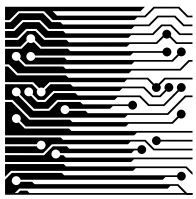
PREDICTIVE HIERARCHICAL LEVEL OF DETAIL OPTIMIZATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF SCIENCE
AT THE UNIVERSITY OF CAPE TOWN
IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Ashton E. W. Mason
July 1999

Supervised by
Edwin H. Blake



Abstract

In this thesis we address *level of detail optimization*, the problem of automatically selecting object detail levels in an interactive visualization. A good selection mechanism should select levels that are appropriate to the viewing situation and the limited time available for rendering. Our principle contribution is the extension of a previous *predictive* approach to cater for hierarchical scene descriptions in which multiple shared representations are provided for groups of objects. This results in savings in rendering and optimization costs and supports the hierarchical nature of typical scenes. We present the first rigorous characterization of the predictive hierarchical level of detail optimization problem, and show its equivalence to a new hierarchical generalization of the *Multiple Choice Knapsack Problem*. This allows us to identify and correct problems with previous approaches.

We present a series of new mathematically proven algorithms in the development of an improved predictive hierarchical level of detail optimization algorithm, including new algorithms for the Hierarchical and conventional Multiple Choice Knapsack Problems. Our level of detail algorithm is *predictive*, guaranteeing that the predicted rendering cost of its selected levels of detail are always lower than the available frame rendering time. It is *hierarchical*, allowing the use of shared group object representations. It is *incremental*, exploiting coherence between successive optimal solutions for increased efficiency. Lastly it is mathematically correct and provides guaranteed levels of predicted perceptual quality. Our algorithm is a significant contribution to the elimination of lag in interactive visualization.

We introduce a new formalism for the investigation and analysis of the hierarchical level of detail problem, the *level of detail graph*. Using them we prove the equivalence of our algorithms, and show how this proof can be adapted to prove the unproven equivalence of previous algorithms. We present the results of a perceptual experiment demonstrating the effectiveness of the use of shared object representations and an implementation demonstrating the practical feasibility of our level of detail optimization algorithm. This represents the first application of hierarchical level of detail optimization to the rendering of scenes generated with hierarchical radiosity.

Acknowledgments

I am grateful to my supervisor Dr. Edwin Blake for his support throughout my degree. His guidance and direction, as well as countless debates and discussions, were vital to my research. In addition he was responsible for several key ideas including the use of the metric that I refer to as “relative value”.

In addition I am indebted to several people for their kind help with areas of this work. Shaun Nirenstein and Simon Winberg performed the experimental work that is described in Chapter 9. I am grateful to them for their input as well as their dedicated and inspired work. Dr. Silvano Martello, Dr. Ulrich Pferschy, Dr. Sven Krumke and Dr. Theo Swart all provided valuable assistance with my work on the Multiple Choice Knapsack Problem. Dr. Tanja van Rij and Dr. Fons Kuijk were instrumental in arranging a working visit to the CWI in Amsterdam which proved to be the turning point of my research. The Institute for Perception Research (IPO) in Eindhoven provided useful information on perceptual experiments. My thanks go to the Foundation for Research Development of South Africa (now the National Research Foundation) for their financial support during my studies.

I wish to thank my parents for their love and support. I dedicate this work to them. Lastly I would like to thank my friends and fellow students for many helpful discussions and many fine games of *BZFlag*.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> "There are a number of challenges that still have to be met before any of these [3D virtual interface] techniques are used routinely in industry. One that I can talk about briefly is time-critical computing, which the networking community calls quality of service. Instead of having algorithms that compute perfectly and take however long they require, we want algorithms that yield a usable result within a given time limit and produce higher-quality results if given more time. This will enable us to schedule the number of frames that we are able to generate and avoid motion sickness. Such time-critical computing requires a new way of looking at algorithms"
> –Andries van Dam, 1996.

Virtual reality and 3D visualization systems suffer chronically from *lag*, the delay between the expected perception of events by the user and their actual perception. Lag is annoying to the user and is associated with motion sickness, deterioration of immersiveness and degradation of user performance. The major source of lag is the complexity of the rendering process, exacerbated by the vast size of typical models, leading to inconsistent and excessive frame rendering times. Many techniques have been proposed to reduce lag, but no software algorithm or advance in rendering hardware has yet succeeded in eliminating it.

In this dissertation we report on work that promises to eliminate a major cause of lag entirely by actively regulating the complexity of the rendered scene model. This constitutes a *time-critical* approach to rendering in which the preservation of consistent and reasonable frame rates is valued above "realism" and spatial image quality. That is not to say that we disregard image quality in

favour of increased speed — rather we seek to control the rendering process actively and intelligently to ensure that the best possible visual perception is achieved without sacrificing consistently adequate frame rates. We believe that the elimination of lag due to excessive rendering times and the regulation of frame rates on which it depends requires a fundamentally different approach to rendering in which the constraints on frame rendering time are explicitly encoded and religiously adhered to, rather than being left to good fortune and the unpredicted whims of the rendering system.

The time-critical approach to rendering embodied by this research has received some attention in the past. However we feel it has been widely neglected until very recently and even now holds nowhere near the important position it deserves. In this dissertation we show that all of the best attempts to investigate it formally, though certainly inspirational, have been flawed by a lack of mathematical rigour that hampers the usefulness of their ideas. By basing our approach more firmly on a sound theoretical foundation, we propose replacement algorithms and new techniques that correct previous problems and elegantly combine two of the most important ideas in contemporary computer graphics: *hierarchical scene descriptions* and *predictive level of detail optimization*. We draw on previous work that shows the advantages of predictive level of detail optimization techniques and shared drawable representations for hierarchical groups of scene objects. We present a new classification of existing level of detail optimization strategies in terms of whether they are predictive and whether they support hierarchical scene descriptions. This classification shows that few predictive strategies have been proposed and, of those, only two have been hierarchical. Our work serves to address this disparity.

We present a formal and general definition of a hierarchical level of detail scene description: a hierarchical scene description that is characterized by the provision of shared representations for groups of related objects. From this general definition we derive rigorous hierarchical versions of intuitive concepts such as levels of detail and the operations and relations associated with them. We distinguish for the first time between the level of detail optimization problems for hierarchical and non-hierarchical scene descriptions. This distinction is driven by the demonstration that the most promising of the predictive level of detail optimization techniques proposed so far is inherently non-hierarchical and does not allow shared representations for groups of scene objects. By considering the implications of shared object representations and clearly outlining their meaning in terms of non-hierarchical scene descriptions, we derive the first formal description of the hierarchical level of detail optimization problem. This allows a rigorous re-evaluation of previous predictive level of detail optimization algorithms and clearly identifies the sources of their limitations.

We develop a new representational tool for the analysis and investigation of the hierarchical

level of detail optimization problem. These *level of detail graphs* are graphical representations of the state spaces generated by hierarchical level of detail descriptions. We use them to recast level of detail optimization as a search problem and, in so doing, prove the equivalence of several level of detail algorithms.

Based on our formal investigation of the hierarchical level of detail optimization problem and analysis of previous approaches we formulate an improved hierarchical predictive level of detail optimization algorithm. This algorithm is truly hierarchical and allows the use of hierarchical scene descriptions with shared representations for groups of objects. Because of its predictive nature it guarantees constant frame rendering times by ensuring that the predicted rendering cost of the selected scene representation is always lower than the available rendering time. Furthermore it actively optimizes the perceptual benefit of the selected representation and produces guaranteed levels of predicted perceptual quality. By virtue of this we correct problems with previous algorithms that made their solutions arbitrarily bad in the worst case. Lastly our algorithm is incremental and so exploits frame-to-frame coherence for improved efficiency by basing its initial solution on the approximate solution found for the previous frame.

Our approach is founded in theory. We prove the correctness and equivalence of our algorithms and develop, where appropriate, formal definitions of concepts introduced. This firm basis in theory allows us to develop more effective techniques. We measure the practical usefulness of our theoretically developed ideas using experimental implementations in working systems. Our first experiment introduces the use of *perceptual evaluation*, the subjective assessment of image sequences by non-expert human users, for establishing a firm connection between the theory and the real world. The second experiment describes the implementation of our predictive hierarchical level of detail optimization algorithm in an actual interactive realtime visualization system. This system allows the exploration of radiosity-generated scene models through the realtime predictive level of detail optimization of thousands of scene objects, and constitutes the first application of hierarchical level of detail optimization to the dynamic view-dependent adaptive rendering of scene models generated using hierarchical radiosity methods.

1.1 Aims

We investigate the implications of hierarchical level of detail descriptions for predictive level of detail optimization. Our aims are:

1. The formal investigation of the predictive hierarchical level of detail optimization problem.

2. The development of formal tools and techniques to aid in this investigation.

3. The development of improved predictive hierarchical level of detail optimization algorithms.

4. The testing of the effectiveness and efficiency of these algorithms.

1.2 Overview

Chapter 2 We begin in Chapter 2 with a review of related work. In that chapter we distinguish between the *hierarchical* and *non-hierarchical level of detail optimization problems*, where the respective problems are characterized by the classes of level of detail models that they allow. Previous level of detail optimization schemes are discussed in light of this distinction. We reflect on an earlier result demonstrating the equivalence of the level of detail optimization problem to the *Multiple Choice Knapsack Problem* (MCKP), and show that this equivalence holds only for non-hierarchical level of detail optimization. We argue that the equivalence assumes the use of a non-hierarchical level of detail description in which no shared representations are provided for groups of objects. The key difference between the problems arises from the fact that in the case of hierarchical scene descriptions with shared representations for groups of objects it is possible to select single shared representations for multiple objects. This difference is the central theme of our research. Finally we review in detail the predictive level of detail optimization algorithms proposed previously and outline their shortcomings. In particular we show that none of the previously proposed algorithms provide guaranteed levels of perceptual quality, in spite of claims to that effect.

Chapter 3 In Chapter 3 we present a greedy algorithm for the MCKP that we prove is half-optimal or better for all instances of the problem. This algorithm is an improvement over the similar algorithm presented by Funkhouser and Séquin, whose solution we show is not guaranteed to be half-optimal as they claim. Our algorithm makes use of a metric, *relative value*, that embodies a useful insight into the nature of the MCKP. Since our aim is the development of level of detail optimization algorithms, we consider the use of this first greedy algorithm in level of detail optimization and find that its most significant limitation in this regard is that it is not *incremental* and performs a complete greedy optimization every time it is applied. With this in mind we also present a second, simplified MCKP greedy algorithm that is capable of being made incremental. This second algorithm is the result of a simplifying assumption that more expensive selections always provide diminishing returns. We show that the simplified algorithm's solution is at least half-optimal for all

instances of the MCKP in which this assumption is true.

Chapter 4 In Chapter 4 we present a formal definition of a general hierarchical level of detail description (or scene model) in which multiple shared representations may be provided for groups of related objects. Along with this we provide formal definitions of concepts such as the partial ordering of levels of detail that such descriptions provide and the incrementation and decrementation operations between them. We define a hierarchical generalization of the MCKP, the *Hierarchical Multiple Choice Knapsack Problem*, to which the hierarchical level of detail optimization problem is shown to be equivalent. This equivalence is demonstrated by means of an intermediatory representation, the *constrained non-hierarchical level of detail description*, in which the hierarchical constraints on level of detail selections implicit in the hierarchical level of detail description are represented explicitly by constraints on the selection of object representations.

Chapter 5 In Chapter 5 we introduce a novel representation, the *level of detail graph*, of the state spaces generated by hierarchical level of detail descriptions. The level of detail graph is a representational tool that allows the formal, visual and semantic analysis of hierarchical level of detail optimization algorithms and the investigation of the hierarchical level of detail optimization problem.

Chapter 6 In Chapter 6 we present a new greedy approximation algorithm for the Hierarchical MCKP. This algorithm is a natural hierarchical extension of our simplified greedy algorithm for the MCKP. Just as that algorithm is half-optimal for a subproblem of the MCKP, so we prove that this algorithm is half-optimal for a subproblem of the Hierarchical MCKP in which more complex selections provide diminishing returns. The extension of the non-hierarchical algorithm makes use of a hierarchical extension of the previously proposed relative value metric to cater for the constraints on selection that are implicit in hierarchical level of detail descriptions.

Chapter 7 Chapter 7 presents a predictive hierarchical level of detail optimization algorithm that is an equivalent incremental version of the greedy approximation algorithm for the Hierarchical MCKP described in Chapter 6. This algorithm is designed to take advantage of frame-to-frame coherence by accepting as an initial solution the approximate solution reached for the previous frame. We prove the equivalence of the incremental and non-incremental algorithms using level of detail graphs.

Chapter 8 In Chapter 8 we describe experimental research involving subjective perceptual evaluation of animation sequences by volunteer users into the effectiveness of hierarchical level of detail optimization and the use of hierarchical level of detail models. A contribution of this experiment is the introduction of *perceptual evaluation* as a means of testing the effectiveness of graphics algorithms.

Chapter 9 In Chapter 9 we present the results of a second experiment, consisting of the implementation and testing of our predictive incremental hierarchical level of detail optimization algorithm in a practical system implemented by students under our supervision. This experiment represents the first application, to our knowledge, of hierarchical level of detail optimization techniques to the interactive realtime rendering of radiosity-generated scene descriptions to provide view-dependent adaptive refinement at render-time. We show that by taking advantage of view-dependent information such as the position and orientation of the viewer, as well as information won during the radiosity computations, it is possible to reduce visible detail in visually unimportant areas of the scene adaptively and dynamically so as to limit frame rendering times without impacting severely on the perceptual benefit of the rendered frames.

Chapter 10 Finally in Chapter 10 we end this dissertation with some concluding remarks and an evaluation of the key results with regard to the aims of the research stated in Section 1.1.

Chapter 2

Background

In this chapter we discuss the related previous work which forms the basis of our research, and in so doing outline more clearly the problem we aim to address. In Section 2.1 we describe the level of detail problem in general form and distinguish between level of detail modeling, optimization and rendering. In Section 2.2 we describe a high level classification of level of detail optimization strategies according to their aims and approaches, showing that the most promising strategies are those that are *predictive*. In Section 2.3 we review previous level of detail optimization strategies in terms of this classification, noting that relatively few are predictive. In Section 2.4 we discuss the advantages of the use of hierarchical scene descriptions and review the extensive use of such descriptions by previous schemes, noting that the number of predictive hierarchical schemes is surprisingly small. In Section 2.5 we review the theory of several variations of the *Knapsack Problem*, a well-known problem in Operations Research. In Section 2.6 we describe a useful equivalence between level of detail optimization and the Multiple Choice Knapsack Problem noted previously by Funkhouser and Séquin, and show that this equivalence is broken by the shared object representations that characterize hierarchical level of detail descriptions. We describe two non-hierarchical predictive level of detail algorithms in some detail, outlining their limitations. Then in Section 2.7 we consider the hierarchical level of detail optimization problem and discuss in detail two hierarchical predictive level of detail algorithms, outlining their limitations in turn. Finally we provide a summary of the main points of the chapter in Section 2.8.

2.1 Introduction to Level of Detail

The technique known as *level of detail* has developed as a means of managing the complexity of rendering at render-time in interactive animation, visualization and virtual reality systems. One aim of such graphics systems is the creation and perpetuation of what we shall refer to as *user conviction*, the extent to which the user is "convinced" of the illusory reality or environment that is being portrayed. Crucial to user conviction are *visual quality* and *temporal quality*. Visual and temporal quality refer to the extent to which the frames (or images) produced by the system and their timing contribute to user conviction. We refer to the temporal quality of the system as *interactivity*, since the timing (or temporal quality) of the frames directly affects the degree to which the user is able to interact effectively with the system. Because both the visual quality and timing of the frames are dependent on the complexity of the rendering process, there is a continual tradeoff between visual quality and interactivity. Any increase in visual quality may be derived at the expense of increased rendering complexity and therefore decreased interactivity, and conversely any improvement in interactivity may be achieved at the expense of decreased image quality. Practical experience and experimental work such as that of Smets and Overbeeke [79] suggest that static resolution factors such as spatial and colour resolution are significantly less important, relatively speaking, for the performance of many interactive tasks than the regulation of a consistent and reasonable frame rate. Therefore care must be taken to ensure that interactivity is not compromised in the quest for more "realism". Level of detail (LoD) techniques make the management of this tradeoff explicit and allow it to be performed dynamically at render-time, whereas previously it was generally hardcoded by the designer of the system at design-time.

The common theme underlying all level of detail techniques is the provision of *multiresolution* representations for scene objects: geometric representations or collections of representations from which distinct drawable representations at a range of detail levels may be extracted. The multiple drawable representations of a given object are referred to as the *levels of detail* or *impostors* [47] of that object (Figure 1). The advantages of multiresolution representations were first pointed out by Clark [16]. Essentially the provision of multiple drawable representations of differing complexities allows the adaptive selection of more appropriate representations for objects in reaction to changing context than is possible with a single fixed representation. This in turn allows the rendering system to control the amount of detail rendered in each part of the scene intelligently and dynamically so as to prevent the rendering of detail that is too fine to be displayed accurately, perceived usefully, and rendered interactively.

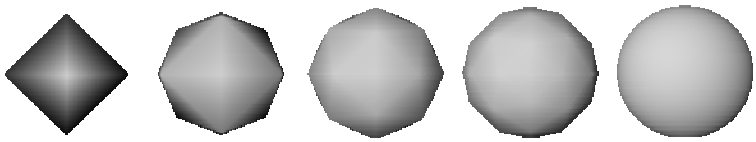


Figure 1: **Levels of detail.** Five representations of the same object (a sphere) at different levels of detail. Each level of detail consists in this case of a different number of planar polygons and has a different rendering cost and a different contribution to the perception of the object as a sphere (and therefore to the perception of the scene as a whole). A typical scene might consist of many different objects, each of which may be provided with its own set of representations at various levels of detail.

We distinguish between *level of detail modeling*, *level of detail optimization* and *level of detail rendering* (see Figure 2). Our interest is focused almost exclusively on level of detail optimization. *Level of detail modeling*, also known as *multiresolution modeling*, is the provision of multiple drawable object representations at various levels of detail. *Level of detail optimization* is the automatic selection of the most appropriate level of detail for each scene object for rendering dynamically at render-time. Finally *level of detail rendering* refers, in this sense, to the rendering of the levels of detail extracted from the level of detail model and selected for rendering by level of detail optimization. To be more precise, level of detail optimization is a process that acts as a filter between the level of detail model and the rendering subsystem, filtering from all the possible representations made available by the model only those that are appropriate for rendering in the current viewing situation. Part of its job is to predict the needs of the user and to adjust its filtering accordingly.

In this dissertation our interest lies primarily in level of detail optimization and in the development of automatic techniques by which it may be performed. We will not address level of detail-specific rendering techniques at all, and our coverage of level of detail modeling will amount to the definition of general-purpose abstract definitions of hierarchical and non-hierarchical level of detail models, with little consideration of the practical techniques by which these models may be implemented in reality. A vast amount of ongoing research has been conducted into the automatic generation and storage of perceptually optimal multiresolution polygonal object representations. Puppo and Scopigno [54] provide a useful survey of these techniques.

While level of detail optimization itself incurs some computational expense, decreasing slightly the time available for actual rendering, this investment is worthwhile since it allows active control over what is rendered. This intelligent management of rendering detail creates the illusion of a far

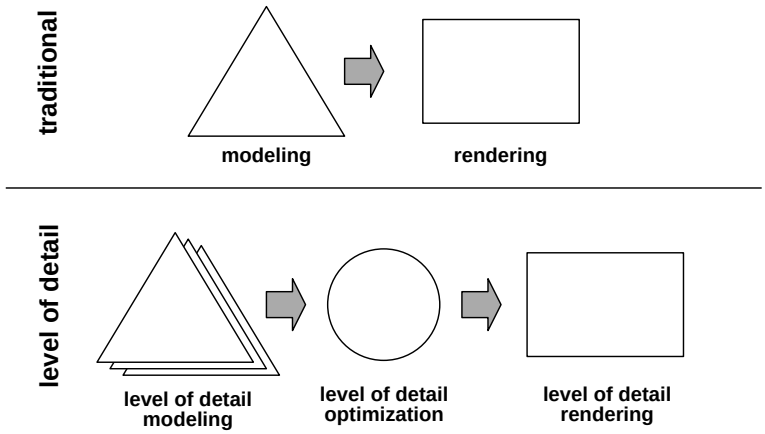


Figure 2: **The level of detail paradigm.** Traditionally the rendering process has been treated as being composed, broadly speaking, of two stages: modeling and rendering. The foundations of the level of detail paradigm lie in *level of detail modeling*, where a single fixed model is replaced by a collection of models of varying complexity. These multiresolution models create a need for *level of detail optimization*, in the form of automatic processes for selecting the particular levels of detail to be rendered, and *level of detail rendering*, the extension of rendering processes to deal with level of detail models. This thesis is concerned with level of detail optimization.

more complex full-detail rendering, by adaptively allocating the rendering time that is available to the detail that will most benefit the perception of the scene. This can, if used properly, create the perception by the user of a constant detail level higher than the greatest constant detail level that could have been rendered if no level of detail optimization were performed (see Figure 3).

2.2 Level of Detail Optimization

Level of detail has been used previously with the following three aims in mind:

1. The prevention of aliasing.
2. The reduction of rendering complexity.
3. The regulation of frame rates.

Aliasing is the misrepresentation of fine (high frequency) detail during the sampling and reconstruction process constituted by rendering [4]. It commonly results in the appearance of jagged

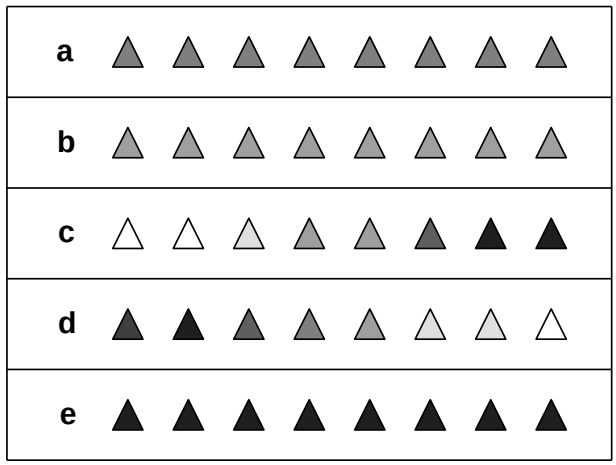


Figure 3: **Why level of detail optimization is worthwhile.** The diagram shows the levels of detail selected for the eight objects of a hypothetical scene, in each of five cases labeled (a) to (e). Levels of detail, or impostors, are represented by triangles, with higher levels of detail being represented by darker triangles. Higher levels of detail have better visual perception but are also more expensive. (a) shows a possible selection with the maximum total rendering cost that can be afforded if no level of detail optimization is performed. (b) shows a selection with the maximum cost that can be afforded after some rendering time is allocated to level of detail optimization. However since level of detail optimization is being performed it is possible to intelligently select, for example, either (c) or (d), which have the same total cost as (b). In each case more important objects are allocated higher levels of detail. Due to the diminished perception of unimportant objects this can create the perception by the user of a constant high level of detail of all objects as shown in (e) with average perceptual benefit greater than that of (a).

lines, random noise and moire patterns in the rendered output, as fine detail is sampled too sparsely and incorrectly reconstructed. By active control over what detail is rendered, it is possible to prevent the rendering of detail that, when rendered, would project to detail too fine to be resolved accurately on the target display. By removing (by means of culling, filtering, smoothing or simplification) any object-space detail that would be reduced by perspective projection to screen-space detail smaller than a pixel it is possible in theory to eliminate aliasing entirely. Furthermore by providing multiple representations at a range of detail levels and selecting between them intelligently it is possible to avoid the rendering of excessive detail (leading to aliasing) on the one hand and the rendering of too little detail (leading to impairment of user conviction) on the other. This amounts to a filtering of detail in object-space, and has historically been put to good use in adaptive and multiresolution

texture mapping techniques such as *mipmapping* [27] [32] as well as in the adaptive rendering of terrain in flight simulators [20]. The use of multiple levels of detail to regulate the display of visual detail was suggested by Clark [16].

Whilst the origins of level of detail lie in perception, it is more commonly associated today with rendering efficiency. A useful side-effect of the removal prior to rendering of detail that may not be correctly displayed is a reduction in the wasted effort of rendering that detail. By adaptive control over the rendering of detail it is possible to significantly reduce the complexity of the rendering process, which is generally worthwhile even at the additional expense of the level of detail selection. Heckbert and Garland provide a useful survey of such techniques [34]. Funkhouser and Séquin, in their broad classification of level of detail approaches, refer to techniques that have this aim as performing *static* detail elision [24]. Static techniques are those that perform detail elision (the removal of unwanted detail prior to rendering) by means of unchanging *static* thresholds, typically based on the distance of objects from the viewer or their projected size on the screen.

Funkhouser and Séquin were the first to note that level of detail techniques are capable not only of reducing the complexity of the rendering process but also of *limiting* it. The aim of limiting the rendering complexity is to guarantee consistent and reasonable frame rates. Experience of real-life interactive visualization systems has shown that a consistent frame rate of at least ten frames per second is essential to the perception by the user of a continuous environment, over and above the need for spatial image quality [79] [63].

The increasing use of advanced clipping and visibility preprocessing techniques that remove invisible model geometry such as those of Greene *et al* [30], Teller and Séquin [80] and Zhang *et al* [88] [89] serves to make the complexity of rendering dependent on the changing complexity of the *visible* portion of the scene, rather than on the constant complexity of the entire scene representation. This tends to cause frame rates to vary dramatically as the user's viewpoint changes to encompass smaller and greater portions of the model. For example, the results reported by Zhang *et al* in [89] show that although the use of *hierarchical occlusion maps* succeeds in culling between 50 and 100 percent of the scene model, its effect on frame rates is to make them *more* variable and irregular than before.

Static level of detail techniques serve to reduce the dependency of the complexity of the rendering process on the complexity of the visible scene by de-emphasizing visually unimportant regions and so reducing the region of the model on which rendering depends to the parts which are *visually important*. Hence static level of detail techniques, while useful for decreasing the dependency of rendering complexity on the complexity of the visible scene, do not remove it completely.

In this sense perceptually-based static detail elision can be seen as an extension of culling to consider not only the coarse binary *yes/no* visibility of objects but also their fuzzy, perceptual visibility. As well as sometimes being culled completely, objects are sometimes replaced with simpler representations. Alternatively, clipping and culling may be seen as coarse specializations of the more general concept of perceptually-based level of detail, in which objects are considered to be either visible (and therefore important) or invisible. Indeed rather than competing with level of detail techniques, advanced culling methods can be used to provide coarse visibility information as a starting point for level of detail optimization.

As the complexity of the visible scene increases, so the complexity of data processed for rendering and hence the complexity of the rendering process increases accordingly. Static techniques are capable of recognizing that an object is only faintly perceived and adjusting its detail level accordingly, but fail to react to the *overall* complexity of the visible scene: the same level of detail will be selected for an object in a certain viewing situation irrespective of whether seven such objects are visible or seven hundred. By reacting intelligently to changing perceptually visible scene complexity it is possible to ensure that no more detail is rendered *in total* than may be rendered in some arbitrary amount of available time. This removes completely the dependency of rendering complexity on the complexity of any part of the scene and ensures (in theory) that the rendering time of every frame is bounded by some fixed upper limit, so that consistent and reasonable frame rates may be achieved. This aim requires more advanced level of detail optimization techniques than those represented by static detail selection: *naively assigning levels of detail based on the distances of objects without any knowledge of global scene complexity is not enough*.

This use of level of detail techniques to bound — rather than merely reduce — rendering times has been referred to as *load balancing* by Reddy [60] and as *time-critical rendering* by Wloka [87], Gossweiler [29] and van Dam [23]. Funkhouser and Séquin distinguish between two paradigms in time-critical level of detail: *adaptive* detail elision and *predictive* detail elision. Adaptive techniques are similar to static techniques, but adjust their detail selection thresholds dynamically according to information garnered from the rendering of previous frames. For example level of detail selection criteria may be made more conservative in response to measurements indicating that the rendering times of preceding frames were excessive, and relaxed in response to measurements indicating that all is well. Funkhouser and Séquin note that these techniques are prone to a tradeoff between rapid reaction speed and stability, leading to *overshoot* (the over-reaction to sudden changes), *oscillation* (the cyclic meandering backwards and forwards across the thresholds) and delayed reaction times [24]. Moreover the very nature of reactive or adaptive techniques is fundamentally limiting: the best

we can hope for is a slight delay in reacting to changes during which the boundaries of acceptable frame generation times are exceeded.[1] These limitations tend to cause adaptive techniques to behave poorly in the regulation of frame rates of complex interactive environments in which the complexity of the visible scene is subject to frequent dramatic and unpredictable changes from one frame to the next [24].

For this reason Funkhouser and Séquin suggest the use of *predictive* level of detail techniques in which the complexity of rendering is limited by means of active prediction of the net effects of the rendering of potential object representations on the total rendering time of each frame. This approach seeks to select for each frame a subset of all possible drawable representations of all scene objects such that their total contribution to user conviction is maximized, while limiting their total rendering cost to some fixed maximum. By limiting the total predicted rendering cost of each frame, it is possible in theory to ensure consistent and reasonable frame rates at the expense of some variation and reduction in spatial image quality. Predictive level of detail optimization is distinguished from static optimization by the fact that it aims to limit rendering complexity *completely* rather than merely reduce it, and from adaptive optimization by the fact that it takes *active* control over rendering complexity rather than passively reacting to previous results. It explicitly predicts and considers the rendering costs of potential representations of individual objects, rather than simply adjusting coarse thresholds. The motivation for performing this level of detail selection dynamically at render-time rather than as a preprocess or during the design of the system is that in an interactive visualization system it is impossible to predict in advance the position and orientation of the viewer's point of view at any given time. Therefore the most interesting parts of the level of detail problem are those that apply to *interactive* visualization systems.

The time-critical approach to rendering embodied by predictive level of detail optimization is closely related to time-critical approaches in other fields. Andries van Dam for example draws an analogy to the concept of guaranteed *quality of service* in networking [23]. Fussel, Read and Silberschatz [56] note that the level of detail concept is general and argue for *system wide multiresolution* — the representation of all data at various levels of detail to allow intelligent management of processing loads. Since the proposal of predictive level of detail for rendering by Funkhouser and Séquin, similar techniques have been applied with some success to the automatic regulation of the accuracy of physical simulation [11], collision detection [38] and update rates of animated objects [87], all with the aim of preserving interactivity at the expense of computational accuracy.

[1] The use of multiple thresholds, known as *hysteresis*, improves the situation slightly by trading detection accuracy for improved stability.

In this dissertation we focus exclusively on predictive level of detail optimization techniques, in the belief that they promise greater success in guaranteeing uniform frame rates in interactive visualization. We feel that static techniques are relatively well established and that the next hurdle in level of detail research is the development of effective and efficient predictive level of detail optimization algorithms. We define the (predictive) *level of detail optimization problem* as follows:

Definition 2.1 *The level of detail optimization problem*

The level of detail optimization problem is the selection of a scene representation for each frame from all available scene representations such that the perceptual benefit of the selected representation is maximized while its total rendering time is limited to some fixed upper bound.

This definition is fairly general and makes no demands nor distinction concerning the form that the scene object's representations may take and how they may be rendered. We assume that efficient level of detail modeling and rendering techniques exist — an assumption that is supported by the large amount of research into multiresolution modeling [54]. We rely on the generality of our approach to ensure that it applies to multiresolution modeling techniques used in practice.

2.3 Previous Level of Detail Optimization Strategies

In this section we review level of detail optimization schemes proposed previously by other researchers, in light of the *static*, *adaptive* and *predictive* classification begun by Funkhouser and Séquin.

The vast majority of previous approaches have focused primarily on level of detail modeling and rendering rather than level of detail optimization, relying on static detail elision for their optimization algorithms. The overwhelming popularity of static techniques is due in part to their simplicity (both in terms of understanding and in terms of implementation) and in part to historical reasons: Static level of detail techniques can be dated back to 1976 [16] whereas predictive methods were first proposed in 1993 [24]. We suspect that while many researchers are aware of the traditional uses of level of detail in preventing aliasing and *reducing* the complexity of rendering, relatively few are aware of its potential use in ensuring consistent and reasonable frame rates. The distinction between *reducing* and *limiting* is subtle and there is a general acceptance of the repeatedly-reinforced "fact" that the complexity of rendering is dependent on the complexity of the visible scene. This perception has begun to change with the recent rise in popularity of *image based rendering*, which

promises to deliver *free* visible scene complexity through the replacement of geometric complexity with images. Image based rendering schemes have varied from replacing realtime rendering of geometry entirely with clever warping of prerecorded imagery to the selective use of images (often derived from renderings of previous frames) to serve as approximate *impostor* representations of scene objects [2] [47] [55] [73] [75]. In the sense of the second usage, image based rendering can be considered a subclass of level of detail based techniques. Both can be considered examples of a general trend towards "faking it" [64] [10] by means of cunning and appropriate approximations rather than brute force computation.

Static detail elision was first proposed by Clark [16]. Clark's aim in proposing the use of multiple levels of detail was perceptually motivated: to guarantee the selection of the most perceptually appropriate level of detail for every object in every conceivable viewing situation. Clark's insight was that by providing multiple explicit representations for each scene object at design-time, it is possible to choose between them at render-time so as to minimize the appearance of insufficiently detailed representations while simultaneously limiting the needless rendering of detail that is too small (due to perspective projection) to be clearly represented or perceived. The most appropriate level of detail in any given situation, in this approach, is the most convincing representation that, when rendered, does not resolve to detail too small to be displayed on the available display.

Because the ability to perceive detail decreases most obviously and predictably with distance from the viewer, the common perception of level of detail among non-specialists has come to be that level of detail selection implies selection by thresholds based on distance. Blake [10] took the idea of perceptually-motivated *user-centric* rendering further, noting that perception of detail depends also on dynamic factors such as the relative motion of objects with respect to the viewer, and providing metrics that predict the most perceptually appropriate level of detail in any situation [9]. Reddy provides a similar perceptually-based framework for rendering [59] [61] [62] and, like Blake, provides a perceptual framework by which the visual effects of rendered detail may be quantitatively measured. We note that although some argument can be made for the existence of perceptual effects that depend on the state of the global visible scene [47], these perceptually-based schemes have assumed for simplicity that the perceptions of independent objects are independent. Therefore perceptually-based level of detail optimization has traditionally been done on a strictly object-by-object basis.

Many level of detail schemes have been proposed that make use of static level of detail optimization in one form or another. Most have followed Clark's lead and based their level of detail selection on simple heuristics that approximate the projected size of objects on the display — either

by some combination of object-space size and distance or by a direct measure of projected screen-space size, and often by considering the object's bounding box rather than the object itself. Rubin and Whitted [67] [68] and Chamberlain *et al* [14] propose schemes in which they aim to prevent aliasing and to reduce rendering times by replacing scene geometry with low detail bounding box representations according to the screen area that they occupy.

Beigbeder and Jahami [6] and Maciel and Shirley [47] propose the use of simple *impostors* such as textured planes as reduced detail representations of groups of scene objects. The Maciel and Shirley scheme is distinguished by the use of a predictive level of detail optimization algorithm, which will be discussed in Section 2.7.1. Shade *et al* [75], Aliaga and Lastra [1] [3] as well as Schaufler *et al* [73] [72] present related schemes in which cached images of previously rendered objects are used as textured impostors to replace distant geometry. Shade *et al* use a perceptually-based error metric that predicts the perceptual difference between the impostor and the full-detail geometry to decide whether or not to use the impostor. Aliaga and Lastra are more concerned with level of detail modeling than optimization, and don't specify how the decision as to whether or not to use an impostor may be made. In [1] and [3] they state that they are investigating how predictive optimization schemes might be used to schedule impostor use automatically so as to guarantee consistent frame rates. In [2] and [55] they describe the adaption of their textured impostor scheme to the simplified representation of *portals* (doorways) between *cells* (rooms) in a densely occluded environment such as a building. In the case of portal textures the selection of geometry for replacement by impostors is dictated by the topology of the building and no attempt is made to place a rigid bound on frame rates. For example the rooms constituting cells may be arbitrarily and non-uniformly complex.

Schaufler and Sturzlinger [73] propose a hierarchical three-dimensional image cache similar to that suggested by Shade *et al*, in which impostor textures are created for each of a hierarchy of nested bounding boxes, and the impostors are substituted for the geometry contained in the bounding boxes they represent if the texels of the impostor textures are smaller than pixels when projected to the display. This constitutes static level of detail optimization based on screen-space area. Schaufler also presents a scheme for the automatic creation of layered texture-based object impostors that lend themselves well to common impostor warping operations [71] [72].

Erikson proposes two related hierarchical polygonal simplification strategies, one with Luebke [45] and the other with Manocha [19]. These schemes are automatic and produce hierarchical simplifications in that they are capable of merging distinct objects. They both operate by collapsing polygon vertices within close visual proximity of each other, and the scheme of Erikson and

Manocha allows the creation of *HLODs* (hierarchical levels of detail), which are shared impostors for groups of objects. These schemes are primarily level of detail modeling strategies, and their level of detail optimization amounts to static detail elision based on screen-space area. No attempt is made to ensure bounded frame rendering times, and while the results presented in [19] show a speedup of up to 10 times over conventional rendering, the speedup depends on the viewpoint of the user and in the worst cases there is no speedup at all. Furthermore the situations when the algorithm's speedup is nonexistent are precisely those in which the conventional rendering takes longest, so that the effect of static level of detail optimization in this case is to *worsen* the non-uniformity of the frame rate rather than improve it.

Luebke and Erikson state in [45] that "the user selects the screen-space size threshold and may adjust it during the course of a viewing session for interactive control over the degree of simplification". This need for user-based control in an effort to maintain reasonable frame rates is a symptom of the lack of awareness of global state that characterizes static detail elision and is exactly what predictive level of detail optimization promises to remove.

Sillion *et al* [78] propose algorithms for the automatic generation of three-dimensional geometric impostors of urban scenery (that is, buildings). They focus on the development of methods by which the scene can be divided into sections that are then represented by simpler impostor representations, and the automatic generation of these impostors. The selection of geometry for dynamic replacement by impostors is based roughly on the distance of the cells (city blocks in this case) from the viewer, and no attempt is made to limit explicitly the total complexity of the selected rendering. As with the rooms of [2] and [55], the city blocks constituting cells may be arbitrarily and non-uniformly complex. It is up to the designer to ensure that they are not.

The *Open Inventor* rendering library by Silicon Graphics provides limited support for level of detail optimization, in the form of special-purpose *SoLevelOfDetail* scene-graph nodes that automatically switch between multiple available representations based on screen-space area thresholds provided by the user [52]. No facility is provided for more advanced level of detail optimization techniques. *VRML*, being closely based on *Inventor*, has similar level of detail support [53] [83]. Level of detail switching is based on the distance of objects from the camera, rather than on their projected screen-space size. Predictive level of detail optimization is difficult or impossible in *VRML*, due to the fact that it is purely a scene-description language.

In addition many techniques have been proposed for the automatic control of the level of detail of artificial terrains [13] [17] [20] [33] [43] [44] (in flight simulators and military-style games, for example). The vast majority of these have made use of some form of hierarchical decomposition

of the landscape data from which custom versions of the terrain may be dynamically extracted. These custom versions are typically adaptively refined according to some error metric that predicts the visual appropriateness of further refinements. These are invariably aimed at eliminating the rendering of detail that, after perspective projection, is too small to be properly resolved. This amounts to static selection based on screen-space area. According to Funkhouser and Séquin [24] some have also attempted to limit frame rendering times adaptively, rather than merely reduce them, by dynamically varying level of detail selection thresholds.

Reddy proposes the extension of level of detail optimization to take into account the portion of the image falling on the fovea, or most sensitive area of the eye [58], and the relative motion of objects with respect to the camera [57]. These ideas were suggested previously by Funkhouser and Séquin [24] and Blake [10] respectively. The degradation of visual detail in the periphery of head-mounted displays has been investigated experimentally by Watson *et al* [84] [85]. These ideas constitute extensions of perceptually driven static level of detail, although they may also be usefully incorporated into predictive level of detail optimization as suggested by Funkhouser and Séquin [24].

Remarkably few researchers have proposed predictive level of detail optimization strategies, since their inception by Funkhouser and Séquin in 1993. The algorithm proposed by Funkhouser and Séquin will be described in detail in Section 2.6.1. The scheme proposed by Maciel and Shirley [47] is an extension of the Funkhouser-Séquin predictive approach to allow the use of hierarchical level of detail models with shared impostors for groups of objects, and will be described in Section 2.7.1. Belblidia *et al* [8] [7] present a predictive hierarchical level of detail optimization algorithm that is similar to but simple than that of Maciel and Shirley. We discuss their approach in Section 2.7.2.

Aliaga *et al* describe the development of a system for rendering massive models; in their case a model of a power station. Their system divides the scene model into a collection of distinct *cells* and directly measures the complexity (in terms of sheer number of polygons) of each cell. It is at least partially predictive in that it then predictively varies the level of the detail selection threshold for each cell so as to limit the total measured complexity of the visible cells. They report that they are able to walk through their 13 million polygon model at interactive frame rates, at the expense of some image quality. Little or no attempt is made to maximize image quality by means of intelligent selection of relative cell detail levels.

Schaufler [70] describes an extension of the scheme of Funkhouser and Séquin which makes use of their level of detail optimization algorithm but combines it with an image caching scheme

similar to those of Shade *et al* [75], Aliaga and Lastra [3], and Schaufler and Sturzlinger [73]. Single complex distant objects are replaced with polygon impostors texture-mapped with dynamically updated images of those objects. Since nothing in Funkhouser and Séquin's original formulation of the algorithm excludes the the use of simple texture mapped impostors rather than conventional polygon-based "levels of detail" this constitutes a combination of the approach of Funkhouser and Séquin with the single-polygon impostor regime, rather than an improvement of the optimization algorithm itself. Indeed the original definition of *impostor* given by Maciel and Shirley [47] explicitly includes both conventional levels of detail and simpler but different styles of representation that only resemble their associated objects visually:

> An impostor is an entity that is faster to draw than the *true object*, but retains the important visual characteristics of the true object. Traditional LODs are a particular application of impostors.[2]

Schaufler notes that the use of image-based impostors saves additional rendering time (over conventional levels of detail) that the Funkhouser-Séquin algorithm, by its predictive nature, is able to apportion intelligently to more important objects. This is of course true for the predictive algorithm of Maciel and Shirley [47] as well, since they also make use of image-based impostors.

Horvitz and Lengyel [37] describe another combination of the predictive Funkhouser-Séquin approach with image based rendering that forms part of the Microsoft *Talisman* rendering system [82]. We describe their approach in more detail in Section 2.6.2.

A survey of contemporary virtual reality solutions (commercial and otherwise) by Reddy in 1995 showed that while roughly half of the systems reviewed supported a means of automatic static level of detail switching (based on distance or screen-space size) only *IRIS Performer* [65] [76] by Silicon Graphics Inc. supported "load balancing", which Reddy defines as automatic modulation of the levels of detail of all objects in an attempt to maintain a fixed frame rate [60]. *Performer* provides traditional distance- or screen-area-based level of detail control. In order to provide frame rate control it also supports adaptive modulation of the thresholds based on estimates of system *stress* derived directly from measurements of rendering *load* (the percentage of the frame time used up by rendering) for the previous frame. The switching thresholds and the modulation by stress and load are individually customizable on an object-by-object basis, providing the ability to favour higher levels of detail for semantically or perceptually important objects. The approach employed

[2] We follow roughly this definition in this dissertation, using *impostor* to denote any useful representation of a scene object of group of scene objects.

by *Performer* is a good example of *adaptive* detail elision as defined by Funkhouser and Séquin [24] (Section 2.2). Rohlf and Helman [65] note that a predictive scheme would be more effective:

> Stress is based on *load*, the fraction of frame time taken to render a frame, and increases as load exceeds a user-specified threshold. The load for frame N is used in conjunction with user-specified parameters to define the stress value for frame $N + 1$, thus defining a feedback network. As discussed in [Funkhouser and Séquin [24]] this method works reasonably well for relatively constant scene densities but suffers because the stress is always a frame late and can exhibit oscillatory behaviour. As illustrated in Figure 9 [not shown], a hysteresis band can reduce stress oscillations but a more sophisticated stress management technique such as that described in [Funkhouser and Séquin [24]] has better characteristics.

OpenGL Optimizer, also by Silicon Graphics Inc., is described as featuring "contribution culling", allowing the developer of the visualization system to create one or more levels of detail by either specifying a target polygon count or polygon count reduction percentage and then having *Optimizer*'s built-in object simplifier build the new levels of detail. A separate pass regroups levels of detail into level of detail nodes. During rendering the selection of the level of detail to render is based on either the viewing distance or the projected area of the model onto the screen [77].

Silicon Graphics describe *Optimizer* as the first "anti-graphics" system, referring to its use of level of detail techniques to actively eliminate geometry from the rendering process. While this is perhaps an overstatement it does reflect the growing acceptance of the idea that the optimal rendering is not necessarily that with the most visible detail.

In concluding this section we note that there is clearly a need for more research into predictive level of detail optimization. Since its inception only a few researchers and practical systems have attempted to use it. In the next section we describe one possible reason why this is so.

2.4 Hierarchical Level of Detail Descriptions

Level of detail optimization techniques have typically made use of *hierarchical level of detail descriptions*, in which objects are grouped hierarchically and shared representations may be provided for groups of objects (see Figure 4). The popularity of hierarchical scene descriptions is due in part to their ability to represent the hierarchical relationships between the various parts of typical scenes, and the fact that *part-whole* hierarchies (in which each object is the union of its children, or parts)

naturally produce scene representations at varying levels of detail. For the sake of comparison we speak also of *non-hierarchical* level of detail descriptions, which are characterized by the provision of multiple representations for objects but not of common shared representations for groups of objects. A scene description might for example be hierarchical in structure (with objects grouped recursively into group objects) but not provide any shared representations for the groups.

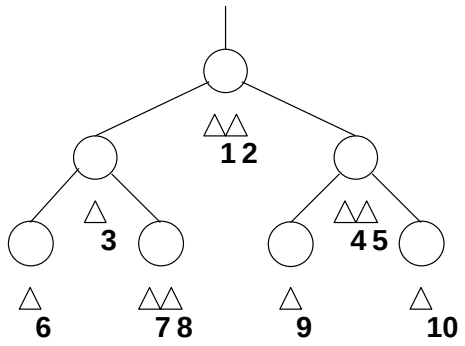


Figure 4: **A simple hierarchical level of detail description.** In this stylized drawing objects are represented by circles and are connected to their children by downward arcs. Each group (non-leaf) object is the union of its children. Impostors, or drawable object representations, are represented by triangles below the objects they represent, in order of increasing detail from left to right. The impostors of group objects (impostors 1 through 5) are effectively shared representations for all of the children of those group objects. For example impostor 5 is a suitable replacement for impostors 9 and 10.

Funkhouser and Séquin claim in [24] that their level of detail optimization algorithm is suitable for application to hierarchical scene descriptions. However we will show in Section 2.6 that this does not extend to catering for shared group representations. This is such a significant limitation that the algorithm of Funkhouser and Séquin cannot really be said to be useful for hierarchical level of detail descriptions. Notably Schaufler presents two separate level of detail schemes, one of which uses Funkhouser and Séquin's predictive optimization algorithm but does not cater for shared group object representations [70] while the other is hierarchical and caters for shared representations but is not predictive and makes no attempt to use a Funkhouser-like scheme [73]. Similarly Horvitz and Lengyel [37] use an approach that is very similar to Funkhouser and Séquin's and is also non-hierarchical with no mention of shared group representations. We discuss their scheme in Section 2.6.2.

Maciel and Shirley, as well as providing a predictive level of detail optimization algorithm, permit the use of truly hierarchical level of detail scene descriptions in which shared representations may be provided for groups of objects [47]. The hierarchical descriptions assumed by Maciel and Shirley is identical in general form to that shown in Figure 4. While the predictive algorithm of Maciel and Shirley is in some sense a hierarchical extension of Funkhouser and Séquin's algorithm it differs significantly from it, as we shall show in Section 2.7.1. Our research constitutes the investigation of ways in which predictive and hierarchical level of detail optimization may be combined into a working whole, and is therefore an extension of the work of both Funkhouser and Séquin and Maciel and Shirley.

Like Maciel and Shirley, Belblidia *et al* [8] allow hierarchical level of detail descriptions with shared object representations. Their level of detail description is identical in form to that of Maciel and Shirley.

Clark was the first to suggest the use of hierarchical level of detail scene descriptions, or hierarchical multiresolution models in which the scene is described in terms of a nested hierarchical decomposition of objects into subobjects or parts and multiple drawable representations may optionally be provided for objects at any level of the hierarchy. Clark's description takes the form of a hierarchy of objects, each of which contains both pointers to other objects which are its children and pointers to other possible representations (or impostors) of that object. Blake [9] assumes the use of a similar structure in his formalization and extension of the perceptually-inspired paradigm of Clark.

Rubin and Whitted [68] make use of an irregular part-whole hierarchical description in the form of a hierarchy of nested arbitrarily oriented rectangular parellalepiped bounding boxes, with no other drawable scene geometry. Only one associated drawable representation is explicitly provided for each object in the hierarchy, however other *implicit* representations are available in the form of the explicit representations of their ancestors and descendants. Note that each non-leaf bounding box functions as a shared impostor for all of the bounding boxes it contains. A complete scene representation may be produced by rendering any subtree of the scene description rooted at the root object.

Chamberlain *et al* [14] present a similar scheme in which a part-whole hierarchical scene representation is automatically generated around the native geometry of the available scene description (Figure 5). They create a regular hierarchical spatial decomposition of the scene in the form of an *octree*, such that the leaves of the octree contain only a limited maximum number of geometric primitives. Each of the non-leaf nodes of the octree is essentially a group object, and has associated

with it a single drawable representation, or impostor. These impostors are colour cubes whose sides are coloured with the average colour of the geometry contained within that cube when viewed from that side, calculated in a preprocessing step by means of orthographic renderings. This scheme differs from that of Rubin and Whitted primarily in that the hierarchical scene description is regular and contains traditional geometry at the leaves as well as bounding boxes.

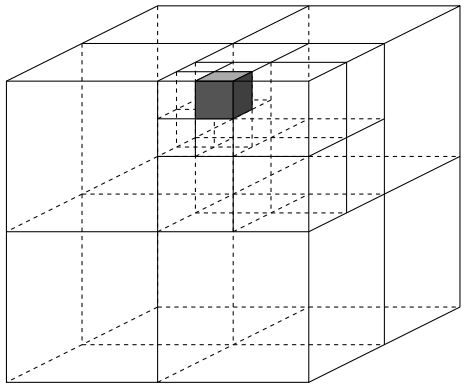


Figure 5: **Hierarchical spatial decomposition in the form of an octree.** An illustration of the manner in which Chamberlain *et al* construct a regular hierarchical spatial decomposition of the scene from an available geometric scene description using an octree.

Marshall *et al* [48] describe another scheme that is similar to that of Chamberlain *et al*, but uses a non-uniform recursive part-whole subdivision of the scene into tetrahedrons instead of cubes. Like the cubes of Chamberlain, these tetrahedrons themselves serve as drawable impostors for their contained geometry.

Wiley *et al* make use of an extended form of the BSP tree called a *Multiresolution Binary Space Partition (MBSP) Tree*, in which multiple representations may be provided for each hierarchically partitioned subspace and a distinct subtree is created for each representation [86]. This represents the conversion of a part-whole hierarchical scene description (the BSP tree) to a hierarchical level of detail description (the MBSP tree).

Shade *et al* [75] present a scheme that is similar in spirit to those of Chamberlain *et al*, Marshall *et al* and Wiley *et al*, but makes use of caching of previously rendered images of scene geometry rather than pre-computed impostors. They use a BSP tree to partition the scene hierarchically, and associate with each node previously rendered images of the scene geometry represented at that node. These previously rendered images are then textured onto billboard-style polygons placed

in the scene to function as impostors for their represented geometry. In this way impostors are generated automatically and cached to improve efficiency by taking advantage of frame-to-frame coherence.

The scheme presented by Schaufler and Sturzlinger [73] is closely related to that of Shade *et al*. They employ a hierarchy of nested bounding boxes, and use textures representing the contents of each bounding box as impostors. This idea also has much in common with that of Chamberlain *et al*, and differs primarily in the use of an irregular bounding box hierarchy rather than a regular octree and the use of cached textures as impostors rather than simple shaded polygons cubes.

Aliaga and Lastra [1] [3] are unusual in that, although they use cached texture-mapped impostors like Shade *et al*, these impostors are not organized in a hierarchy. Their work is concerned more with the low level technicalities of using single-polygon texture-mapped impostors (for example, accounting for the visual errors introduced by parellax) than with level of detail optimization. While hierarchical level of detail descriptions are certainly useful for level of detail modeling, much of their benefit arises in level of detail optimization where their recursive nature and power of expression allows efficient ways of dealing with the complexity of large scenes. Chamberlain *et al*'s use of an octree for example allows them to easily select simple shared representations (cube impostors) for large groups of related objects in one step.

The scheme proposed by Sillion *et al* [78] for the automatic generation of 3D geometric impostors of urban scenery can be considered to be essentially hierarchical. They partition the model into cells corresponding loosely to city blocks, and create impostors that represent entire cells or combinations of cells.

The ideas presented by Erikson and Manocha [19] and Luebke and Erikson [45] combine elements of traditional polygonal simplification (or multiresolution modeling) and hierarchical level of detail descriptions. Of the two, Erikson and Manocha's scene model is closer to a regular hierarchical level of detail description in which objects are grouped recursively and dynamically generated shared representations are provided for the groups. The scene model of Luebke and Erikson is not as regular and consists of a *vertex tree*: a hierarchical listing of all the vertices in the scene, according to proximity and without regard to object topology, that may be queried on a frame-by-frame basis to provide custom scene representations in which no groups of distinct vertices are closer together than a given screen-space threshold.

One disadvantage of this vertex tree structure from a level of detail modeling point of view is that it discards useful information about the scene topology that might otherwise be used to ensure that the collapsing of vertices resulted in visually meaningful simplifications of groups of

objects. Secondly it loses the advantages of the use of *display lists* (in *OpenGL*, for example) that fixed[3] levels of detail such as those assumed by Funkhouser and Séquin allow. In particular the dynamic creation of custom impostors favours the use of current graphics hardware in *immediate mode* rather than *display list mode*, since unique impostors are generated for every frame and there is therefore little advantage to be gained from display lists. The use of immediate mode often causes a reduction in speed by a factor between 2 and 3 [45]. On the other hand the dynamic polygon-level simplification of geometry allows the system of Luebke and Erikson to provide view-dependent simplifications of individual objects in which parts of large objects which are near to the viewer are simplified less than parts that are further away. Hoppe [36] presents a related technique for view-dependent adaptive refinement of mesh-based object representations.

Ultimately the distinction between "fixed" and "dynamic" impostors comes down to the level of the geometry at which the objects one considers to be available for manipulation by level of detail optimization are located. Luebke and Erikson regard individual polygons as available for simplification and decimation, while more traditional approaches [16] [9] [24] have assumed that all polygon-tweaking is performed as a pre-process in the generation of a relatively small number of fixed drawable representations. Clearly the level of geometry which one regards as "objects" dictates what can and can't be done with the objects one chooses. The work we present here is inspired by the orthodox view of scene-object-level impostors and so our discussion will be formulated with this in mind. However our approach is general enough to be applied at any level of geometry and in fact in Chapter 9 we present an experiment in which polygons are considered individually by our level of detail optimization algorithm. But as Chapter 9 shows, our experience suggests that while the perceptual benefits of optimizing individual polygons may be great, the performance benefits of level of detail optimization may be lost in the overhead that this involves. As a general rule we suggest that, for the purposes of efficiency, the cost of considering whether or not to render an impostor should be significantly lower than the cost of simply rendering that impostor.

As we said in Section 2.2, the many techniques that have been proposed for the automatic control of the level of detail of artificial terrains typically make use of some form of hierarchical decomposition of the landscape data. Often this is in the form of a quadtree, or regular 2-dimensional decomposition [20]. In the same way that the faces of the cubes in Chamberlain *et al*'s octree hierarchy serve as impostors for the geometry contained in those cubes, so the faces of the quadtree nodes serve as impostors for their contained geometry. In these systems 2-dimensional hierarchical

[3] Fixed levels of detail are also sometimes called *static* levels of detail to distinguish them from the dynamic creation of impostors at render-time that systems such as those of Luebke, Erikson and Manocha allow. We avoid this term, preferring to use it to distinguish between static, adaptive and predictive level of detail optimization.

level of detail descriptions are exploited for the same benefits as 3-dimensional ones: the automatic generation of simplified impostor representations for groups of objects and the efficiency of being able to consider entire groups of objects at the same time in level of detail optimization.

IRIS Performer [76] allows the use of hierarchically nested level of detail nodes in its scene description hierarchy. Using these nodes, multiple level of detail representations may be provided for the parts of larger level of detail representations. This style of hierarchical level of detail description is similar to but slightly more flexible than that of Maciel and Shirley [47], depicted earlier in Figure 4. In this thesis we assume a Maciel-Shirley style hierarchical description, which we describe formally in Chapter 4.

All of the hierarchical level of detail schemes discussed above are non-predictive (See Section 2.2), with the notable exceptions of those of Maciel and Shirley [47] and Belblidia *et al* [8]. The non-predictive schemes make no attempt to place a firm restriction on rendering complexity and are designed to reduce, rather than bound, frame rendering times and (in some cases) to ensure perceptually appropriate rendering. The complexity of the rendered geometry is still firmly linked to the complexity of the visible portion of the scene, although in many instances the severity of the dependency is reduced. As we noted in Section 2.3 the only predictive methods proposed so far (to our knowledge) are the algorithm of Funkhouser and Séquin [24], the level of detail system of Schaufler [70], which simply makes direct use of Funkhouser and Séquin's algorithm, and the hierarchical level of detail optimization schemes of Maciel and Shirley and Belblidia *et al*. This suggests, firstly, that predictive level of detail optimization is a difficult problem in need of further research (given that the schemes that exist have not completely solved it) and secondly that the incorporation of hierarchical level of detail descriptions into predictive level of detail is still an open problem. The only proposals which have addressed it, to our knowledge, are those of Maciel and Shirley and Belblidia *et al*. Their schemes apply a predictive approach inspired by Funkhouser and Séquin to hierarchical level of detail descriptions. We will investigate them in more depth in Sections 2.7.1 and 2.7.2.

2.5 Knapsack Problems

In their seminal predictive level of detail optimization paper [24] Funkhouser and Séquin note that the level of detail optimization problem is equivalent to the *Multiple Choice Knapsack Problem*. For this reason we provide in this section a brief overview of the Knapsack Problem and several of its variations. In addition we describe several approximation algorithms for these problems. We will

return to the question of level of detail optimization in Section 2.6.

The *Knapsack Problem* describes a class of real-life optimization problems in which a limited subset of a set of *candidate items* must be selected, such that their total profit is maximized while their total cost is limited. Each item has a certain constant *cost* and *profit*. The problem draws its name from an analogy to the filling of a knapsack of limited capacity with a selected number of available items, each of which has a particular intrinsic worth (profit) and a certain size or weight (cost). Many variations on the theme have been discussed at length in the field known as Operations Research. We outline a few of them here.

2.5.1 Binary Knapsack Problem

The *Binary Knapsack Problem* (0-1 KP) is a specialization of the Knapsack Problem in which each item may be selected exactly zero or one times (as opposed to the more general problems in which multiple or fractional selections are permitted). 0-1 KP is shown conceptually in Figure 6 and may be defined formally as follows:

Definition 2.2 *The Binary Knapsack Problem*

Given a set N *of* n candidate items *and a* knapsack, *with*

$$\begin{aligned} p_j &= \text{profit } \textit{of item } j \\ w_j &= \text{cost } \textit{of item } j \\ c &= \text{capacity } \textit{of the knapsack} \end{aligned}$$

maximize

$$z = \sum_{j=1}^{n} p_j x_j$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} w_j x_j &\leq c \\ x_j \in \{0,1\} \quad \forall \quad j &\in N \end{aligned}$$

where

$$x_j = \begin{cases} 1 & \textit{if item } j \textit{ is selected} \\ 0 & \textit{otherwise} \end{cases}$$

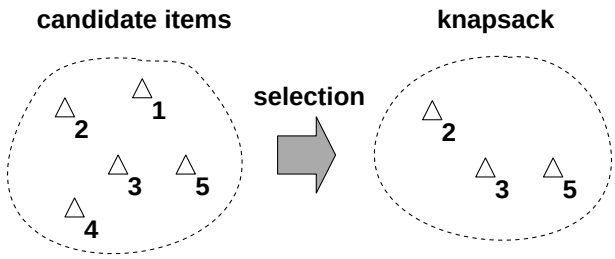


Figure 6: **The Binary Knapsack Problem (0-1 KP).** A variation of the Knapsack Problem in which each candidate item may be selected only zero or one times. Each item has a certain *cost* and *profit*, and the object is to select the subset of the candidate items that maximizes the total profit and has total cost lower than the capacity of the *knapsack*.

2.5.2 Multiple Choice Knapsack Problem

The *Multiple Choice Knapsack Problem* (MCKP) is a generalization of 0-1 KP in which the candidate items are partitioned into a collection of distinct *candidate subsets* (or types) and exactly one item must be selected from each subset. It is shown conceptually in Figure 7. The MCKP will serve as the basis of our work in later chapters, due to its usefulness as a model of the level of detail optimization problem.

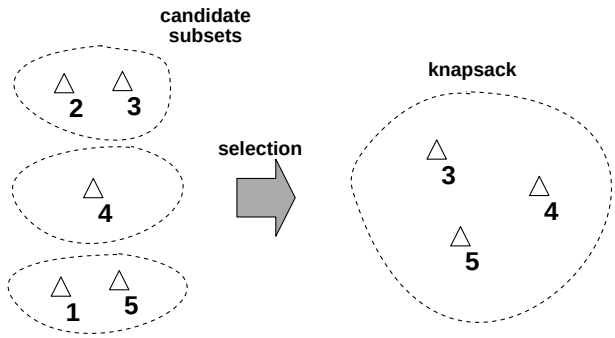


Figure 7: **The Multiple Choice Knapsack Problem (MCKP).** The problem consists of a set of *candidate items*, partitioned into a collection of *candidate subsets*. Each item has a certain constant *cost* and *profit*. The object is to select the subset of the candidate items with greatest total profit, subject to the constraints that exactly one item must be selected from each candidate subset and the total cost of the selected items must be less than or equal to the capacity of the *knapsack*.

Definition 2.3 *The Multiple Choice Knapsack Problem*

Given a set N of n candidate items*, a partition into disjoint* candidate subsets $N_1, \ldots, N_r$ *of the item set N and a* knapsack*, with*

$$\begin{aligned} p_j &= \text{profit } \textit{of item } j && (1)\\ w_j &= \text{cost } \textit{of item } j && (2)\\ c &= \text{capacity } \textit{of the knapsack} && (3) \end{aligned}$$

maximize

$$z = \sum_{j=1}^{n} p_j x_j \tag{4}$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} w_j x_j &\leq c && (5)\\ \sum_{j \in N_k} x_j = 1 \quad \forall \quad k &\in \{1, \ldots, r\} && (6)\\ x_j \in \{0,1\} \quad \forall \quad j &\in N && (7)\\ N = \{1, \ldots, n\} &= \bigcup_{k=1}^{r} N_k && (8) \end{aligned}$$

assuming

$$N_h \cap N_k = \emptyset \ \forall \ h \neq k. \tag{9}$$

The *Continuous Multiple Choice Knapsack Problem*, C(MCKP), is a relaxation of MCKP in which the candidate items may be fractionally selected (as opposed to either completely selected or not selected at all, as in the MCKP). It is also known as the *linear* or *continuous relaxation* of MCKP. Although C(MCKP) is not central to our work, we include it here because an understanding of it is useful in later chapters.

Definition 2.4 *The Continuous Multiple Choice Knapsack Problem*

Given a set N of n candidate items*, a partition into disjoint* candidate subsets $N_1, \ldots, N_r$ *of the item set N and a* knapsack*, with*

$$\begin{aligned} p_j &= \text{profit } \textit{of item } j\\ w_j &= \text{cost } \textit{of item } j\\ c &= \text{capacity } \textit{of the knapsack} \end{aligned}$$

maximize

$$z = \sum_{j=1}^{n} p_j x_j$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} w_j x_j &\leq c \\ \sum_{j \in N_k} x_j = 1 &\quad \forall \quad k \in \{1, \ldots, r\} \\ 0 \leq x_j \leq 1 &\quad \forall \quad j \in N \\ N &= \{1, \ldots, n\} = \bigcup_{k=1}^{r} N_k \end{aligned}$$

assuming

$$N_h \cap N_k = \emptyset \ \forall \ h \neq k.$$

Intuitively, instead of selecting exactly one item from each candidate subset exactly once, we must select a *total* of exactly one item from each candidate subset, possibly composed of fractional portions of several items.[4]

2.5.3 Algorithms for 0-1 KP

The 0-1 KP is NP-complete, but may be solved in psuedo-polynomial time by *dynamic programming* techniques. These are efficient for small problem instances. For larger instances *branch-and-bound* algorithms are generally used [49]. Branch-and-bound techniques are examples of algorithms that "while acknowledging the apparent inevitability of exponential time complexity, seek to obtain as much improvement over straightforward exhaustive search as possible" [25]. They construct a search tree of possible partial solutions and utilize powerful bounding methods to recognize partial solutions that cannot possibly be extended to actual solutions, thereby eliminating entire branches of the search in a single step. While their worst case performance approaches the complexity of an exhaustive search, their average behaviour for typical problems is acceptable.

Aside from algorithms for the optimal solution of 0-1 KP, there are several *approximation* algorithms that provide solutions of guaranteed near-optimal quality in exchange for polynomial time

[4] Our definition of C(MCKP) is identical to that presented by Martello and Toth [49]. Garey and Johnson [25] use *Continuous Multiple Choice Knapsack* to refer to a different NP-complete problem in which exactly one distinct item must be selected from each subset, but this item may be fractionally selected with any selection value ranging from zero to one. Note that this does not imply that a total of exactly one item is selected from each subset.

complexity. Sahni [69] and others have described fully polynomial time approximation schemes for 0-1 KP that consist, essentially, of a series of approximation algorithms of increasing polynomial time complexity that provide approximate solutions of any arbitrary non-optimal guaranteed quality.

A simple *greedy* approximation algorithm for 0-1 KP is shown in Figure 8. This algorithm corresponds roughly to the least expensive algorithm in Sahni's approximation scheme and is guaranteed to produce a solution with total profit at least half as good as the profit of the optimal solution (we say that it is *half-optimal*). Its time complexity is $O(n \log n)$ [49].

input: an instance of the 0-1 KP (see Definition 2.2)
output: a half-optimal or better solution to the instance

// consider items in descending order of value,
// selecting each item if we can afford to

begin
set $s \leftarrow 0, x_i \leftarrow 0 \; \forall \; i \in N$
order the set N of candidate items by descending value (p_i/w_i)
for each candidate item j
{
if $\sum_{i \in N} x_i w_i + w_j \leq c$ **then**
set $x_j \leftarrow 1$
else
if $s = 0$ **then**
set $s \leftarrow j$
}

// consider the critical item solution

if $s \neq 0$ **then**
if $p_s > \sum_{i \in N} x_i p_i$ **then**
set $x_i \leftarrow \begin{cases} 1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \quad \forall \; i \in N$
end

Figure 8: **Greedy approximation algorithm for 0-1 KP.**

The greedy algorithm performs a *greedy selection* of the candidate items using a *heuristic*, or simple guideline, that suggests which item should be selected next. The heuristic used is to

consider at each stage the available item with the greatest *value* (profit / cost). Value provides a measure of the "bank for the buck" provided by prospective representations. When each item is considered it is placed in the knapsack if there is sufficient remaining space and discarded if there is not. The rationale behind this heuristic is that whenever an item is considered we know that all of the remaining unconsidered items have lower value and therefore would provide an inferior use of the space taken up by it. Notice that because items may only be either completely selected or completely unselected, this generally results in some wasted unused space. The greedy selection therefore works as long as the selection of an item does not prevent the selection of a large later item of lower value that, due to the specific sizes of the items, would have provided a more profitable filling of the knapsack. To guard against this pathology the algorithm compares the profit of the solution resulting from the greedy selection stage to the profit of the solution consisting of only the *critical item*, which is the first item to be discarded during greedy selection due to its cost being greater than the remaining space in the knapsack. If the *critical item solution* has greater profit the algorithm rejects the greedy solution and selects the critical item instead. This step resolves the pathology and guarantees that the algorithm's solution is at least half as good as the optimal solution (in terms of total profit).

We present here a proof of the half-optimality of the 0-1 KP greedy algorithm, based in part on ideas presented in [49]. We include it here only because in Chapters 3 and 6 we will extend it to provide proofs of the half-optimality of our own algorithms. The basic approach underlying the proof is that of relating the solution value of the optimal solution to that of the algorithm's approximate solution, producing an expression for the maximum error or difference between the two. By considering the implications of the selection order of items we then reduce this maximum error expression to terms involving only weights and the value of the critical item. We then formulate another expression relating the weights of the maximum error equation to the weight of the critical item and, substituting, show that the maximum error is bounded by the profit of the critical item. Since the greedy algorithm also considers the solution consisting of only the critical item, we deduce that the algorithm's solution is at least half as good as the optimal one.

Theorem 2.1 *The total profit of the 0-1 KP greedy algorithm's solution is at least half as good as the total profit of the optimal solution, for every instance of 0-1 KP.*

Proof:

Assume that the candidate items have been ordered by descending value, such that

$$\frac{p_i}{w_i} \geq \frac{p_j}{w_j} \quad \forall \quad i < j \in N.$$

Let G be the set of items selected by the greedy algorithm before the critical item s is considered. Then since s is the first item to be rejected the total profit z^g of the items in G is given by

$$z^g = \sum_{i<s} p_i.$$

The profit z of the optimal solution is related to z^g by

$$z = z^g + \sum_{i \in Q} p_i - \sum_{i \in R} p_i$$

where Q is the set of items that are in the optimal solution but not in G, and R is the set of items that are in G but not in the optimal solution. Then since

$$\frac{p_i}{w_i} \geq \frac{p_s}{w_s} \quad \forall \quad i \in G$$

and

$$\frac{p_i}{w_i} \leq \frac{p_s}{w_s} \quad \forall \quad i \notin G$$

we know that

$$\frac{p_i}{w_i} \geq \frac{p_s}{w_s} \quad \forall \quad i \in R$$

and

$$\frac{p_i}{w_i} \leq \frac{p_s}{w_s} \quad \forall \quad i \in Q$$

and so

$$z \leq z^g + \sum_{i \in Q} w_i \frac{p_s}{w_s} - \sum_{i \in R} w_i \frac{p_s}{w_s} \tag{10}$$

$$\leq z^g + \left[\sum_{i \in Q} w_i - \sum_{i \in R} w_i \right] \frac{p_s}{w_s}. \tag{11}$$

Now we know that

$$\sum_{i \in Q} w_i - \sum_{i \in R} w_i \leq \overline{c}$$

where $\overline{c}$ is the space left in the knapsack after the selection of G and before the rejection of s, since $\{G \cup Q\} - R$ fit into the knapsack by definition.

Since s was rejected we know that $\overline{c} \leq w_s$. Hence

$$\sum_{i \in Q} w_i - \sum_{i \in R} w_i \quad \leq \quad w_s$$

and so, from (11),

$$\begin{aligned} z &\leq z^g + w_s \frac{p_s}{w_s} && (12) \\ &\leq z^g + p_s. && (13) \end{aligned}$$

Recall that the greedy algorithm compares the total profit of the final greedy solution (which is greater than or equal to z^g) to the profit p_s of the critical item solution, and keeps whichever solution is better. That is, the algorithm's solution has profit $z^h \geq max(z^g, p_s)$. Therefore, from (13)

$$z^h \quad \geq \quad \frac{1}{2} z$$

and the profit of the algorithm's solution is guaranteed to be at least half the profit of the optimal solution.

2.5.4 Algorithms for MCKP

The MCKP, like 0-1 KP, is NP-complete. However it too may be solved by branch-and-bound algorithms and in psuedo-polynomial time by dynamic programming techniques [49]. Branch-and-bound algorithms for MCKP are preceded by a *reduction* phase, in which items that are *dominated* by other items (and therefore are not elements of an optimal solution) are removed. The complexity of the reduction phase is $O(n \log m)$, where m is the cardinality (or size) of the largest candidate subset [49]. Then at each stage of the branch-and-bound algorithm an instance of the Continuous Multiple Choice Knapsack Problem, C(MCKP), is constructed and solved. Algorithms for the exact solution of C(MCKP) are presented for example by Armstrong *et al* in [5] and by Dudzinski and Walukiewicz in [18].

We however are particularly interested in *approximation* algorithms for MCKP, rather than optimal algorithms. There are several reasons for this. Most obviously, the approach taken by Funkhouser and Séquin [24] is based on a greedy algorithm. More importantly, optimal algorithms are invariably more expensive than approximation algorithms for the same problem. Since our chosen

application is level of detail selection in computer graphics, where approximation of visual quality in exchange for reasonable execution speeds is the norm, we do not require optimal solutions to our NP-problems and are happy to settle for approximate solutions of guaranteed quality levels in return for relatively efficient time complexities. This favouring of guaranteed execution speed over completely accurate solutions is in keeping with our time-critical computing-style approach.

Lastly, and most importantly, the greedy approximation algorithms that we propose may be made *incremental*, so that they accept as input an initial solution derived from the approximate solution found for the previous problem instance. This allows us to exploit the coherence between the problem instances presented by successive frames, in a manner that was originally proposed by Funkhouser and Séquin [24].

Funkhouser and Séquin present a greedy algorithm for a variation of C(MCKP), which they use as a basis for an incremental level of detail optimization algorithm (described in Section 2.6.1). In effect they use this algorithm as an approximation algorithm for MCKP, since, as we note in Section 2.6, their level of detail optimization approach assumes distinct unitary object representations. The greedy algorithm presented by Funkhouser and Séquin has a time complexity of $O(n \log n)$. They claim that it is *half-optimal*: that its solution is guaranteed to be at least half as good as the optimal solution in terms of total profit. However we shall show that this is not the case.

The Funkhouser-Séquin algorithm is shown in Figure 9. Like the 0-1 KP algorithm presented earlier, the algorithm operates by considering items for selection in descending order of *value* (profit / cost), selecting them if they can be afforded. When an item is considered that belongs to the same candidate subset as an item already in the knapsack, only the item with greater profit is retained. That is, if the new item has greater profit then it replaces the currently selected one. Otherwise it is discarded. The thinking behind this is that the retained item, whichever it is, represents a better use of the space it takes up than any of the unconsidered lower-valued items [24].

It is not clear for *which* problem the Funkhouser-Séquin greedy algorithm is intended to be an approximation algorithm. Funkhouser and Séquin state that it is an approximation algorithm for the "Continuous Multiple Choice Knapsack Problem", and refer to two sources: *Computers and Intractability* by Garey and Johnson [25] and a paper called *The Multiple Choice Knapsack Problem* by Ibaraki *et al* [39]. However these two sources define the Continuous Multiple Choice Knapsack Problem differently. In [25] it is defined as a relaxation of the Multiple Choice Knapsack Problem in which exactly one distinct item must be selected from each subset, but this item may be fractionally selected with any selection value ranging from zero to one. In [39] Ibaraki *et al* deal with the continuous relaxation of MCKP which we (as well as Martello and Toth [49]) call C(MCKP),

input: an instance of the MCKP (see Definition 2.3) or C(MCKP)
output: a solution to that instance

```
        // consider items in decreasing order of value, selecting items if we
        // can afford them and replacing items from the same candidate
        // subset on higher profit

begin
    set x_i ← 0 ∀ i ∈ N
    order the set N of candidate items by decreasing value (p_i/w_i)
    for each candidate item j
    {
        if there exists an item l such that x_l = 1 and l, j ∈ N_k for some k then
            if p_j > p_l and Σ_{i∈N} x_i w_i − w_l + w_j ≤ c then
                set x_l ← 0, x_j ← 1
        else
            if Σ_{i∈N} x_i w_i + w_j ≤ c then
                set x_j ← 1
    }
end
```

Figure 9: **The Funkhouser and Séquin greedy algorithm.**

defined previously in Definition 2.4. These problems are significantly different, although both are NP-complete. Finally, as we note in Section 2.6, Funkhouser and Séquin actually use their algorithm as an approximation algorithm for the conventional MCKP.[5]

No matter which of these problems the Funkhouser-Séquin algorithm is intended for, it is flawed. In the case of both MCKP and C(MCKP), the algorithm fails to guarantee that its solution is *feasible*: that (a total of) exactly one item is selected from every candidate subset. Consider for example what happens in the case where, after the candidate items are ordered by descending value, all of the items from one candidate subset are considered only after the available knapsack capacity has been completely spent.[6]

[5] It is important not to be confused by the fact that the MCKP and even its "continuous" relaxation C(MCKP) are often defined such that the profits and costs of each item are integer. This is a convenience and is assumed without loss of generality. In practice the profits and costs can be any positive real numbers [49]. Consider for example that any instance of MCKP represented with finite accuracy on a computer can be converted to an equivalent instance containing only integral profits and costs by multiplying through by a constant factor. The word "continuous" in "continuous relaxation" refers to the selection values x_i, rather than the profits p_i and costs w_i.

[6] In the case of the variation of the Continuous Multiple Choice Knapsack Problem defined by Garey and Johnson

More importantly, Funkhouser and Séquin claim that their algorithm is guaranteed to produce a solution with total profit that is at least half as good as the total profit of the optimal solution. It is however possible to find two kinds of counterexample for this, each of which outlines a separate problem. An example of the first type of counterexample, shown in Definition 2.5, is an instance of the MCKP in which there are two candidate subsets (N_1 and N_2), each of which contains only one item. N_1 contains item 1, and N_2 contains item 2. Item 1 has profit $p_1 = 100$ and cost $w_1 = 1$ and item 2 has profit $p_2 = 9999$ and cost $w_2 = 100$. The capacity of the knapsack is 100. The algorithm selects item 1, while the optimal solution consists of item 2.

Definition 2.5 *First counterexample for the Funkhouser-Séquin greedy algorithm*

Consider the instance of the MCKP in which:

$$\begin{aligned} \overline{p} &= (100, 9999) \\ \overline{w} &= (1, 100) \\ r &= 2 \\ N_1 &= \{1\} \\ N_2 &= \{2\} \\ c &= 100 \end{aligned}$$

The solution reached by the greedy algorithm for MCKP in this instance is

$$\overline{x} = (1, 0)$$

with a total profit $z^g = 100$*, while the optimal solution is*

$$\overline{x} = (0, 1)$$

with a total profit $z = 9999$*. Therefore the solution reached by the Funkhouser-Séquin greedy approximation algorithm in this instance is less than half as good as the optimal solution.*

Note that this counterexample is effective for both MCKP and C(MCKP), since C(MCKP) is a relaxation of MCKP. It may be made arbitrarily bad simply by changing the profits of the items. It represents the pathological case in which the selection of an item prevents the selection of a later item that in fact happens to provide a much better usage of that particular knapsack capacity. This

[25], this is still a feasible solution, since in their problem it is not required that exactly one item is selected in total from each candidate subset.

problem is inherent in the naive greedy approach and occurs in the similar greedy algorithm for the 0-1 KP (see Section 2.5.1). It may be corrected, as it is in the case of the 0-1 KP algorithm, by simply comparing the result of the greedy selection against a solution containing the *critical item*, which is the first item to be discarded due to insufficient remaining space in the knapsack, and taking whichever is better. [7]

However the Funkhouser and Séquin greedy algorithm also suffers from another intrinsic and more serious limitation:

Definition 2.6 *Second counterexample for the Funkhouser-Séquin greedy algorithm*

Consider the instance of the MCKP in which:

$$\begin{aligned} \overline{p} &= (10, 900, 910, 10, 600, 10, 400) \\ \overline{w} &= (1, 100, 700, 1, 500, 1, 400) \\ N_1 &= \{1, 2, 3\} \\ N_2 &= \{4, 5\} \\ N_3 &= \{6, 7\} \\ c &= 1000 \end{aligned}$$

The solution reached by the Funkhouser-Séquin greedy algorithm in this instance is

$$\overline{x} = (0, 0, 1, 1, 0, 1, 0)$$

with a total profit $z^g = 930$*, while the optimal solution is*

$$\overline{x} = (0, 1, 0, 0, 1, 0, 1)$$

with a total profit $z = 1900$*. Therefore the solution reached by the algorithm in this instance is less than half as good as the optimal solution.*

Note that this counterexample, like the first, is valid for both the MCKP and its continuous relaxation. Unlike the first, it cannot be corrected simply by comparing the greedy solution against the critical item solution. The critical item in this instance is item 5, and its profit is 600. The critical item solution, after augmentation with the lowest cost items from the other candidate subsets to ensure feasibility, is $(1, 0, 0, 0, 1, 1, 0)$, with a total profit of 620. This counterexample therefore

[7] In order to construct a feasible critical item solution we might augment the critical item with the lowest cost items from every other candidate subset.

represents a pathology that does not afflict the greedy algorithm for the 0-1 KP, and is a result of the additional constraint imposed on the MCKP and its relaxations that we may select at most one item from each candidate subset.

Conceptually, the second counterexample represents the pathological case in which an item with high profit and low cost is replaced by another from the same candidate subset with slightly higher profit but much higher cost (and correspondingly much lower value). The large gain in total cost incurred for the small gain in profit prevents the algorithm from selecting later items (including the critical item) that, together with the replaced item, would have resulted in a much higher solution value. The act of replacement is more risky than the act of simple selection since it implies the loss of the replaced item, which may potentially provide a high profit for a low cost. By replacing we may sacrifice a large gain in cost for a small gain in profit.

In this instance the erroneous replacement is the replacement of item 2 with item 3. Notice that item 3 has slightly higher profit than item 2, but much greater cost. However the value 1.3 of item 3 is still higher than the value 1.2 of item 5. The optimal solution actually consists of the critical item 5 and the replaced item 2, together with item 7. Note however that the profit of the replacing item is always greater than the profit of the replaced item (since the algorithm only replaces if the net profit increases). Therefore the extra penalty incurred by replacements, of losing a possibly high valued item in exchange for a lower valued item with higher profit, is only really a penalty if the increased cost prevents more desirable later items from being selected. If later items are discarded due to there being insufficient available space for their selection then by definition the critical item must be one of them. In Chapter 3 we propose an approximation algorithm for the MCKP that accurately predicts the effects of the replacement of items and so corrects the faults of the Funkhouser-Séquin algorithm.

We note that other authors have proposed several correct approximation algorithms and approximation schemes for MCKP. Chandra *et al* [15] for example present a fully polynomial time approximation scheme. Their scheme contains an algorithm that is half-optimal and whose time complexity is $O(nr \log n)$, where r is the number of candidate subsets. Lawler [42] presents another approximation scheme, based on that of Ibarra and Kim [40]. Its complexity is $O(n \log n + \frac{rn}{\epsilon})$, where ϵ is an accuracy measure $0 < \epsilon \leq 1$ such that the error of the approximation is bounded by a factor ϵ of the optimal solution. Gens and Levner [26] present an approximation algorithm based on approximate binary search whose solution is at least $\frac{4}{5}$ as good as the optimal solution and whose time complexity is $O(n \log r)$. While these approximation algorithms are efficient, they may not, as far as we know, easily be made incremental. In Chapter 3 we present a greedy algorithm for MCKP

that forms the basis for the level of detail optimization algorithms that we describe in later chapters. Our algorithm may readily be made incremental and has a time complexity of $O(n(r+m))$, where m is the cardinality, or size, of the largest candidate subset. Although it is simple, we can find no precedent for it in the literature.

2.6 Non-Hierarchical Level of Detail Optimization

In this thesis we distinguish between the *non-hierarchical* and *hierarchical* level of detail optimization problems. These problems are the application of the predictive level of detail optimization problem defined in Section 2.2 to non-hierarchical and hierarchical level of detail descriptions, respectively. Although this might not seem obvious, the two are fundamentally different and the particular approach to solving the predictive level of detail optimization problem suggested by Funkhouser and Séquin [24] is, as it stands, *only applicable to the non-hierarchical problem*. This has important implications for the predictive hierarchical level of detail algorithm for Maciel and Shirley [47], which is essentially an attempt to extend the predictive approach of Funkhouser and Séquin to the hierarchical level of detail optimization problem.

Funkhouser and Séquin's approach to solving the level of detail optimization problem is, broadly speaking, to perform a cost-benefit analysis of the rendering of each possible representation of each scene object. By considering the *perceptual benefit* (predicted contribution to the perception of the scene) and *rendering cost* of each object representation, they aim to select the scene representation that provides the best possible contribution to the perception of the scene without taking longer to render than some fixed maximum permissable rendering time. They assume that there are a finite collection of distinct objects and that each object has a finite collection of distinct impostors or drawable representations which constitute its levels of detail. Noting that it is always desirable to select exactly one representation for every visible object they show that under these assumptions the level of detail optimization problem is equivalent to the Multiple Choice Knapsack Problem. The level of detail optimization algorithm that Funkhouser and Séquin propose is an incremental version of a greedy algorithm for the MCKP.

Funkhouser and Séquin actually state that the level of detail optimization problem is equivalent to the *Continuous Multiple Choice Knapsack Problem* [24]. However we argue that it is better modeled by the MCKP itself. In either case, predictive level of detail optimization is the selection of a subset of a set of available candidate object representations, subject to constraints on the selection of objects and their total cost. An instance of the level of detail optimization problem is reformulated

as an instance of the Multiple Choice Knapsack Problem (see Definition 2.3) as follows:

1. Each *candidate item* corresponds to one object impostor.

2. The *cost* and *profit* of each item correspond to the rendering cost and perceptual benefit of its associated impostor.

3. The *capacity* of the knapsack corresponds to the available frame rendering time. The total rendering cost of the selected scene representation must be less than or equal to this capacity.

4. The candidate items are partitioned into disjoint *candidate subsets* that correspond to the objects to which the impostors belong. The selection of one item from a candidate subset corresponds to the selection of one impostor for its associated object.

5. The objective of the problem is the maximization of the total profit of the selected items. This corresponds to the aim of maximizing the perceptual benefit of the impostors selected for rendering.

The constraints on the selection of impostors that distinguish MCKP from C(MCKP) are of vital importance. In asserting the equivalence of level of detail optimization to the *Continuous* Multiple Choice Knapsack Problem, Funkhouser and Séquin allow for the selection of non-integral portions of candidate items (and therefore incomplete object impostors). However in their level of detail optimization they treat object impostors as distinct units that may only be either completely selected or not selected at all [24]. We suggest that the level of detail optimization problem tackled by Funkhouser and Séquin is better characterized by MCKP itself (in which candidate items may only be either completely selected or not selected at all) than by its continuous relaxation C(MCKP).

We can easily envisage a variation of the level of detail optimization problem in which object impostors can be partially rendered; for example that arising from the use of *progressive mesh* object representations from which an impostor at any given continuous level of detail may be extracted dynamically at render-time, as proposed by Hoppe [35]. Instead of a collection of distinct alternate representations of each object, a progressive *meta-representation* may be provided for each object. This makes it possible to select essentially any fractional proportion of the full detail rendering of each object (up to the finite resolution of the progressive mesh), inviting comparisons to a continuous relaxation of MCKP such as C(MCKP). Indeed if only one progressive meta-representation is provided for each object (as seems sensible) then the problem is no longer multiple-choice and reduces to the linear relaxation of 0-1 KP, for which effective greedy algorithms exist. However

Funkhouser and Séquin, in our opinion, assume traditional disjoint representations that cannot be fractionally rendered [24].

Furthermore we foresee a complication with the continuous relaxation-based approach. The nature of the continuous relaxations of 0-1 KP and MCKP imply that the cost and profit resulting from the selection of an item are exactly proportional to the selection value of that item. That is, if the selection value of a given item i is x_i, then the profit resulting from it is $x_i p_i$ and the cost is $x_i w_i$. In practice progressively more complex renderings of objects typically provide *diminishing returns*: proportionally smaller increases in perceptual benefit for further increases in rendering cost. For example the perceptual benefit of a rendering of an object with 100000 polygons is not typically 100 times greater than the perceptual benefit of another rendering with only 1000 polygons. In general the perceptual benefit of a representation is related to its rendering cost by some complicated arbitrary function. The linear relaxations of 0-1 KP and MCKP are fundamentally incapable of representing this non-linearity, being linear optimization problems. The MCKP, with its multiple distinct candidate items in each candidate subset, is capable of coping with the non-linearity by approximation. The multiple items in each candidate subset of the MCKP serve essentially as a piece-wise approximation or sampling of the cost and benefit functions of some continuous range of possible representations, as shown in Figure 10.

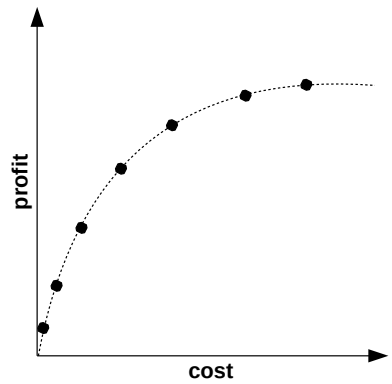


Figure 10: **Approximation of perceptual benefit function by multiple candidate items.** The items in a candidate subset, shown as points on a graph of increasing profit vs. increasing cost. Note how the multiple items in a candidate subset serve as approximations of some actual continuous function describing the returns, in terms of perceptual benefit, provided by a series of progressively more complex renderings.

It is worth noting that even a progressive mesh is capable of producing only a finite number

of different object impostors. A progressive mesh therefore really functions as a compact or compressed store of a large but finite number of distinct impostors. An important advantage of progressive meshes however is that they can be *non-uniformly* generated to favour higher and lower levels of detail in different areas of a single object, according to the current viewing situation [36]. There is therefore no single meaningful ordering of the potentially generatable impostors in terms of increasing level of detail. In this research we ignore progressive meshes and assume multiple distinct object representations. We note however that the multiple representations of varying perceptual benefit and rendering cost that a progressive mesh provides might usefully be represented by the multiple candidate items in each candidate subset of the MCKP.

In this work we will take it as given that the predictive non-hierarchical level of detail optimization problem (as outlined by Funkhouser and Séquin) is equivalent to the MCKP. We assume that each object impostor may be either completely selected or not selected at all, and that exactly one impostor representation must be selected for each object. The objective is to maximize the total perceptual benefit of the selected representations while limiting their total rendering cost. Funkhouser and Séquin exploit this equivalence to produce a non-hierarchical level of detail optimization algorithm based on their greedy algorithm for MCKP described previously in Section 2.5. We describe their level of detail algorithm in Sections 2.6.1.

Of course, the definition of the MCKP assumes that the profit and cost of each item is well defined. In practice *perfectly accurate* numerical measures of the perceptual benefit and rendering cost of arbitrary object representations are tricky or even impossible to define, as Funkhouser and Séquin note [24]. The situation isn't helped at all by the fact that to be useful in realtime level of detail optimization, these measures must be calculated efficiently for large numbers of potential scene object representations. Traditionally, simple heuristic metrics for perceptual benefit have been defined that attempt to provide a flavour of the perceptual issues involved (for example, the attenuation of vision over distance by perspective and atmospheric haze and the limitations of human perception) for a small investment of computational cost. Blake [10] and Reddy [59] [62] have been chief among these. Likewise Funkhouser and Séquin propose the use of simple heuristics called *benefit* and *cost* to predict roughly the perceptual benefit and rendering cost of object representations as required. They suggest various factors (similar to those mentioned by Blake and Reddy) that such heuristics might ideally take into account. In practice even the heuristics used by Funkhouser and Séquin are as rudimentary as possible and take into account only the distance of the object from the viewer and simple estimates of the perceptual accuracy of impostors (in the case of benefit) and empirically-derived estimates of their relative rendering cost (in the case of cost). In this thesis we

ignore entirely the development of more useful heuristics and restrict our attentions to level of detail optimization algorithms themselves. We assume that useful (although probably not completely accurate) benefit and cost heuristics may always be defined.

The equivalence between the level of detail optimization problem and the MCKP is subject to an important assumption that results from the fact that in the MCKP there is no concept of *dependencies* between the selected representations of different objects. It assumes that the selection of the representations of separate objects is independent; that selecting a given representation for one object neither forces nor prevents the selection of any representation of any other object. This assumption holds for the problem tackled by Funkhouser and Séquin (due to their implicit assumption that all objects and their representations are distinct) but is broken by hierarchical scene descriptions in which shared representations may be provided for groups of objects. We refer to the class of scene descriptions in which multiple representations may be provided for objects but no *shared* representations may be provided for groups of objects as *non-hierarchical level of detail descriptions*. Such a description is illustrated in Figure 11.

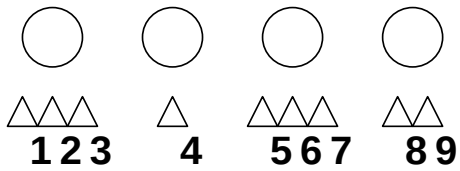


Figure 11: **Non-hierarchical level of detail description.** A simple non-hierarchical scene description with multiple levels of detail. The scene consists of a collection of distinct *objects* (shown conceptually as circles). Multiple drawable representations, or *impostors*, may be provided for each object (shown as triangles). Impostors are numbered and the impostors of each object are shown in ascending order of detail from left to right. Compare with Figure 4 and note that no *shared* representations may be provided for groups of objects.

The selection of a shared group representation for one object implies the selection of that representation for all of its siblings, as shown in Figure 12. We therefore refer to the subset of the level of detail optimization problem that Funkhouser and Séquin handle (by virtue of their use of the equivalence to MCKP) as the *non-hierarchical level of detail optimization problem*. The provision of shared representations for groups of objects that characterizes hierarchical level of detail descriptions invalidates this, giving rise to the *hierarchical level of detail optimization problem*. We shall discuss this further in Section 2.7.1.

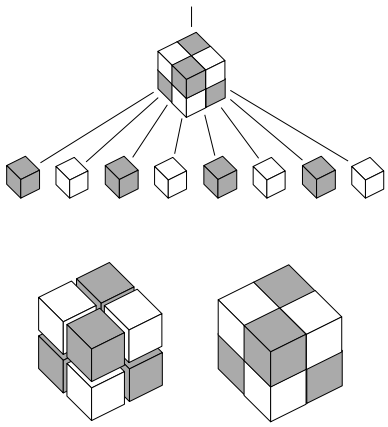


Figure 12: **Implications of shared object representations.** The provision of a shared representation for a group of objects introduces the implicit constraint that when the shared representation is selected for one object it must be selected for all of them. In this example, a single shared cube representation (on the right) is provided for the eight individual cube objects on the left.

The equivalence between level of detail optimization and the MCKP is subject to another important assumption: it assumes that the predicted perceptual benefit (or profit) and rendering cost (or cost) of each representation is independent of the representations selected for other objects. Maciel and Shirley note in [47] (and more extensively in [46]) that in practice this is not generally the case. The selection of a particular representation for one object may influence the perception and rendering cost of other representations of other objects. In the field of psychology the dependence of the perception of one object on the perception of another is an aspect of *Gestalt perception* [28]. It is illustrated by an example in Figure 13. Foley *et al* [22] list the Gestalt *rules of group perception* that predict the perception of groups of objects, with particular regard to user interface design [22]. Although Maciel and Shirley draw attention to this assumption they choose explicitly to ignore it [47]. Like Maciel and Shirley we satisfy ourselves with representing only the dependencies between objects that are children of the same group object, ignoring more complex and subtle dependencies and focusing on what can be deduced about the hierarchical level of detail optimization problem by assuming that the equivalence between the non-hierarchical problem and the MCKP holds.

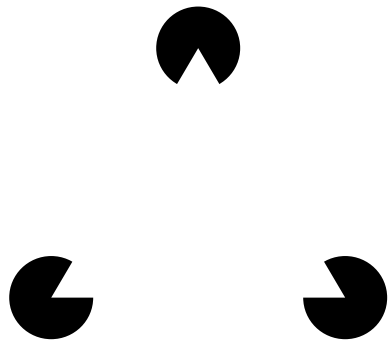


Figure 13: **Example of Gestalt perception.** Each black shape may be perceived as a circle overlaid with one corner of a white triangle, because of the presence of the other two black shapes. If any of the black shapes is removed then the perception of the triangle and therefore of the other shapes is changed (after [28]). Furthermore if one of the shapes were to be approximated by a low detail impostor consisting of a complete circle then the perception of the others would also change.

2.6.1 Funkhouser-Séquin Algorithm

Since the MCKP is equivalent to the non-hierarchical level of detail optimization problem (as discussed in the previous section), the greedy approximation algorithm for the MCKP proposed by Funkhouser and Séquin (described in Section 2.5.4) is also an approximation algorithm for the non-hierarchical level of detail optimization problem. Indeed, it is for this purpose that it was proposed. However a naive implementation of this algorithm as a level of detail optimization algorithm would require calculating and sorting the benefits, costs and values of all object representations in every frame (since they are dependent on the viewing situation and so generally change from one frame to the next). Funkhouser and Séquin therefore formulate an incremental version of the algorithm that bases its initial solution on the approximate solution found for the previous frame, taking advantage of frame-to-frame coherence in the form of the similarity of the optimal solutions of consecutive frames. This algorithm is their *predictive incremental non-hierarchical level of detail optimization algorithm*, and embodies the constrained optimization approach that later inspired the predictive hierarchical algorithm of Maciel and Shirley, which we describe in Section 2.7.1.

Funkhouser and Séquin claim that the incremental algorithm is equivalent to the MCKP greedy algorithm for a restricted subproblem of the MCKP in which the *values* (profit / cost) of items within each candidate subset always decrease uniformly as their cost increases [24]. In terms of level of detail optimization, this implies that higher (more expensive) levels of detail of objects

should always provide *diminishing returns*. If this requirement is satisfied then the two algorithms are equivalent, otherwise the action of the incremental algorithm is uncertain. By definition higher levels of detail have higher rendering cost and greater perceptual benefit. It is worth noting that this approach assumes that the levels of detail of each object are subject to a constant ordering by ascending perceptual benefit and rendering cost.

The algorithm is designed to be applied once before the rendering of each frame, and its output is a set of detail levels for the scene objects that is used to decide which object representations should be rendered. The advantage of the incremental algorithm over the original greedy algorithm is purely one of efficiency: it exploits the typically large coherence between successive frames (and therefore their optimal detail levels) by accepting as input an initial solution that is generally the solution found for the previous frame. The algorithm, shown in Figure 14, is iterative and operates by successively improving the initial set of selected impostors by a series of refinements, which are referred to as *incrementations* and *decrementations*. Each incrementation and decrementation consists of the replacement of the selected impostor of a single object with its immediately higher or lower detail impostor, respectively. Generally a large number of possible incrementations and decrementations is available at any stage. The algorithm chooses those that best serve to maximize the total value of the selected set, by eliminating (by decrementation) representations with low value and selecting (by incrementation) those with high value. This strategy works because of the assumed ordering of levels of detail by ascending cost and descending value.

In each iteration of the algorithm the selected impostor of one object is incremented. Then the selected impostors of other objects are decremented, while the total rendering cost of the selected solution is higher than the available frame rendering time. In each case the object chosen for incrementation is that whose *subsequently* selected impostor (after the potential incrementation) has *greatest* value. When an object is chosen for decrementation it is that whose *currently* selected impostor has *lowest* value. In this way the algorithm, over the course of several iterations, tends to use up as much of the available rendering time as possible while favouring the higher levels of detail of objects whose higher levels of detail provide better "bang for the buck" [24]. In this way it attempts to ensure that the best possible scene representation is provided while limiting the total rendering cost to the permitted maximum.

The algorithm terminates when the impostor selected during incrementation in an iteration is immediately deselected by a decrementation in the same iteration. When this occurs the algorithm has reached its final approximate solution and there is no further work for it to do. After termination of the algorithm the selected impostor of each object is scheduled for rendering. For simplicity

```
input: an instance of the non-hierarchical level of detail optimization problem
input: a feasible initial solution to that instance
output: an improved solution to the instance

        // iteratively improve the selected representation by incrementing
        // the selected impostors of some objects and decrementing others

begin
    select the impostors of the provided initial solution
    set done ← FALSE
    while done = FALSE
    {

        // increment the selected impostor of one object

        if any objects are not at their highest levels of detail then
        {
            find the object o which is not at its highest level of detail
            and whose immediately higher impostor has greatest value
            increment the selected impostor of o
        }

        // decrement one object, while the total rendering cost is too high

        while the total cost is greater than the rendering cost limit
        {
            find the object p which is not at its lowest level of detail
            and whose selected impostor has lowest value
            decrement the selected impostor of p
            if p = o then
                set done ← TRUE
        }
    }
end
```

Figure 14: **The Funkhouser-Séquin incremental level of detail algorithm.**

the algorithm as shown here assumes that the rendering cost limit is sufficient to select at least the lowest levels of detail of all objects, but not so great that the highest levels of detail of all objects may be selected. The special cases in which these assumptions do not hold may be weeded out by trivial tests before the algorithm begins. In such cases the lowest and highest levels of detail of all objects are selected, respectively.

Equivalence

Funkhouser and Séquin state the equivalence of their greedy and incremental algorithms without proof in [24]. We provide a proof of the equivalence of two related algorithms in Chapter 7 that can be easily adapted to serve as a proof for the equivalence of the Funkhouser and Séquin algorithms.[8] The equivalence claim can be intuitively supported by noting that the criterion by which objects are selected for incrementation is similar to that by which items are selected for replacement in the greedy algorithm: in both cases the item (or impostor) whose immediately higher cost impostor has greatest value is selected for replacement. The criterion for decrementation in the incremental algorithm is the inverse of that for incrementation: the impostor selected for replacement by its immediately lower cost impostor is that whose value is lowest. To understand why the *decreasing value* assumption is necessary, note that the incremental algorithm only ever moves one "step" at a time: always only considering the immediately higher and lower level of detail of each object. It is because higher levels of detail always have lower value that it is safe to do this; we know that a poor immediately higher detail impostor with no hope of selection can never hide another higher detail impostor of the same object with greater value.

The greedy algorithm is not guaranteed to produce a *feasible* solution in which exactly one item is selected from every candidate subset, while the incremental algorithm is guaranteed to select exactly one impostor for each item (even in the case where the available rendering time or knapsack capacity is too low). The equivalence of the algorithms therefore depends on the assumption that either the greedy algorithm is forced to select exactly one item from each subset (by preselecting the lowest cost items from each one) or the problem instance is augmented with an imaginary "null item" in each subset with negligible cost and profit.

[8] Since we have shown that the algorithm of Funkhouser and Sequin is invalid, and since we present a better algorithm in Chapter 3, we do not provide this proof.

Complexity

The time complexity of the Funkhouser and Séquin incremental algorithm is $O(n \log n)$ in the worst case where n is the number of impostors (or items), like its non-incremental counterpart. This is due to the fact that in the worst case we must consider and select all object impostors, resulting in $O(n)$ iterations. In each iteration one incrementation and possibly several decrementations take place. The number of decrementations per iteration is typically small and may be assumed to be $O(1)$ (In the worst case it is n, but in that case the maximum number of algorithm iterations is 1). Each incrementation or decrementation involves the selection of the highest subsequently valued or lowest currently valued impostor, which is $O(1)$ due to the use of ordered priority queues [24]. The updating of the queues after each incrementation or decrementation is optimally $O(\log n)$.

However since the algorithm is incremental and bases its initial solution on the solution found for the previous frame, it only performs a small number of iterations on average. The average complexity is therefore better than $O(n \log n)$. Situations in which significantly more iterations are performed are those in which the initial solution is significantly different from the final solution. For example the first frame, and frames in which the viewing direction is dramatically different from the frame before. Funkhouser and Séquin do not discuss or address the computation times of their algorithm in such cases. In Chapter 9 we report on an experiment investigating the practical behaviour of our own incremental level of detail algorithm, the results of which can also be applied to the Funkhouser-Séquin algorithm.

Optimality

Funkhouser and Séquin claim that their level of detail optimization algorithm, being equivalent to their greedy algorithm for MCKP for a restricted subproblem of the MCKP, produces a solution that is always at least half as good as the optimal solution (in terms of total profit or predicted perceptual benefit) for that subproblem. Recall however that in Section 2.5.4 we provided a counterexample for which the greedy algorithm's solution to MCKP (and, for that matter, C(MCKP)), is not half-optimal. That counterexample (refer to Definition 2.6), is also a counterexample for the incremental level of detail algorithm, for the subproblem in which items within candidate subsets are ordered by ascending cost and descending value. Therefore the Funkhouser and Séquin incremental level of detail optimization algorithm is not half-optimal for that subproblem as they claim.

A level of detail interpretation of the counterexample is a scene in which an object (candidate subset 1 in the example) has one representation with high perceptual benefit and very low cost (item

2), and another with slightly higher perceptual benefit but much higher cost (item 3). For example, the first representation could be a texture map that is inexpensive to render but is visually very similar to the second representation consisting of hundreds of polygons. The algorithm is in danger of inappropriately incrementing the level of detail of the object from the cheap to the expensive representation, thereby excluding the less "valuable" representations of other objects.

2.6.2 Horvitz-Lengyel Algorithm

Horvitz and Lengyel [37] present another predictive level of detail optimization scheme that is closely related to and clearly inspired by that of Funkhouser-Séquin. Essentially it, like that of Schaufler [70] mentioned in Section 2.3, represents the application of the knapsack problem-based constrained optimization approach to an image-based rendering system in which texture-mapped polygon impostors (or *sprites*) are substituted for actual geometry. The images mapped onto the impostors are object representations rendered from the original geometry in previous frames, warped by affine transformations that approximate the actual perspective and geometric transformations caused by the changes in the position and orientation of objects relative to the camera from one frame to the next. The advantage of using the impostors is that it is substantially cheaper to warp and render an existing image-based representation than to re-render the original geometry. A predictive optimization algorithm is used to decide for each object at the start of each frame whether to substitute the current image-based impostor or re-render the geometry. Heuristics similar to the benefit and cost heuristics of Funkhouser and Séquin predict the "perceptual cost" of using an impostor and the extra rendering cost incurred by re-rendering the actual geometric representation of an object. Perceptual cost is essentially the inverse of perceptual benefit and represents the view that using impostors detracts from the "perfect" perception of the full detail representation.

The scheme of Horvitz and Lengyel differs from the Funkhouser-Séquin and Schaufler schemes in that their optimization algorithm is based on the greedy approximation algorithm for 0-1 KP described in Section 2.5.1 rather than on an algorithm for MCKP. Horvitz and Lengyel make the simplifying assumption that each object has only two possible representations: the full detail geometry and the image-based impostor. Then since each (visible) object must be represented by at least its image-based impostor, the decision for each object amounts to the selection or non-selection of the full detail re-render. This presents an optimization problem that is equivalent to the 0-1 KP, in which the profits and costs of the "items" are the perceptual costs of the low-detail impostors and the extra rendering costs of the high detail representations. The aim is to select the subset of the items with greatest total profit (or lowest total perceptual cost) while limiting the total extra rendering cost

to the available re-rendering time.

Because of this assumption and the resulting equivalence to 0-1 KP rather than MCKP Horvitz and Lengyel are able to use the well-known greedy algorithm for 0-1 KP. They order the candidate items (expensive re-renderings) by descending value (profit / cost) and select as many as they can with the rendering time available. They then compare this greedy solution against the solution consisting of only the item with greatest profit, which is a simpler form of the critical item solution test (see Section 2.5.1). As we proved in Section 2.5.1, this algorithm always produces a solution that is guaranteed to be at least as half as good as the optimal one. Its time complexity is $O(n \log n)$. Horvitz and Lengyel use the greedy algorithm as it is, rather than formulating an incremental version, so unlike Funkhouser and Séquin they fail to take advantage of frame-to-frame coherence and must perform an entire greedy optimization over all scene objects for every frame.

In order to use the 0-1 KP greedy algorithm, Horvitz and Lengyel must assume that every object has only two possible representations. In a typical system in which there are multiple levels of detail for each object (and possibly in multiple dimensions) this assumption does not hold. To cope with this difficulty they suggest a simple approach in which detail reductions in other dimensions (such as reduction of texture resolution or geometric detail) are evaluated by calculating the cost so saved and considering the selection of additional re-renders that this allows. No guarantees exist on the quality of solutions reached using this approach.

2.7 Hierarchical Level of Detail Optimization

From our survey of previous level of detail approaches in Sections 2.3 and 2.4 it is apparent that although both predictive level of detail optimization and hierarchical level of detail optimization are desirable, to our knowledge few predictive strategies have been proposed and only two of these are hierarchical. These two predictive hierarchical level of detail optimization algorithms are those of Maciel and Shirley [47] and Belblidia *et al* [8].

In Section 2.6 we pointed out that the equivalence between the predictive level of detail optimization problem and the Multiple Choice Knapsack Problem noted by Funkhouser and Séquin is dependent on several crucial assumptions, one of which is invalidated by hierarchical level of detail descriptions in which shared representations are provided for groups of objects. We therefore distinguished between the hierarchical and non-hierarchical level of detail optimization problems. Here we show that the predictive optimization algorithm of Funkhouser and Séquin, being essentially a greedy algorithm for the MCKP, is inherently non-hierarchical. The key difference between

the problems arises from the fact that in the case of hierarchical descriptions it is possible to select single *shared* representations for multiple objects. This difference is the central theme of this thesis, and our aim is to investigate the hierarchical level of detail optimization problem and formulate improved optimization algorithms for it.

In response to the difficulties posed by shared group representations, one might suggest simply doing away with them completely. However shared representations for groups of objects accomplish more than would separate representations for those objects of identical visual complexity. Shared representations provide several advantages that are crucial to many of the level of detail optimization strategies reviewed in Section 2.4 (for example, those of Chamblerlain *et al* [14] and Shade *et al* [75]) and a great number of level of detail models:

1. As we noted in Section 2.4, they allow increased computational efficiency since multiple objects can be dealt with by the consideration of only one shared impostor.

2. With shared simple representations it is possible to ensure that visual and topological coherence is maintained between the simplified representations of related objects. For example in Figure 13 it might be useful to ensure that simple representations for the three black shapes are consistent in their representation of the three shapes. Likewise it might be desirable that a group of related objects are either all textured-mapped or not texture-mapped at all. The sharing of low detail representations between objects serves to *constrain* their possible selection, ensuring that inappropriate combinations of low and high detail representations are not selected.

3. Shared geometric representations are generally more storage and computationally efficient than collections of separate representations for the same objects that are visually equivalent, due to the elimination of redundant geometry and the ability to make use of simplified topologies.

These benefits account partly for the pervasiveness of hierarchical representations of various forms in everyday life. Hierarchical decompositions aid the design of complex systems such as massively integrated circuits where each component is assembled from coherent subcomponents whose function is abstracted from their design.

In this Section we discuss the two predictive hierarchical level of detail algorithms that we are aware of: that of Maciel and Shirley and that of Belblidia *et al*.

2.7.1 Maciel-Shirley Algorithm

Maciel and Shirley claim in [47] that their level of detail optimization algorithm is a hierarchical extension of the approach of Funkhouser and Séquin, re-iterating Funkhouser and Séquin's overstatement that the level of detail optimization problem is equivalent to the MCKP. In actual fact the Maciel-Shirley algorithm differs significantly from the Funkhouser-Séquin algorithm, as we might expect.

Maciel and Shirley assume a hierarchical level of detail description such as that shown previously in Figure 4. This description is a *part-whole* description and is distinguished from that of Funkhouser and Séquin by the fact that objects are grouped recursively and multiple shared representations, or *impostors*, are provided for groups of objects. At least one impostor must be provided for every object.

Maciel and Shirley define heuristic measures called *importance* and *accuracy* that predict the inherent importance of objects to user conviction and the visual accuracy of object impostors. Importance is based on the identities of objects and their position and size on the screen, and is independent of their selected representation. It is defined principally for leaf objects (those without children), and the importance of non-leaf objects is assumed to be the maximum of the importance measures of their children. Accuracy predicts the visual "appropriateness" of impostors to a given viewing situation and depends for example on impostor complexity and the direction from which they are viewed.

In addition a *cost* heuristic predicts the rendering cost of impostors. The aim of the algorithm can be expressed simply in terms of these measures: to limit the total cost of the selected scene representation while maximizing its total perceptual benefit (which is assumed to be the sum of the importance and accuracy measures of the selected object representations).

The algorithm is applied once per frame and consists of two stages. In the first stage, the problem of having multiple impostors explicitly associated with each object is solved by traversing the tree and selecting, for each object in turn, one of its associated impostors to serve as its *available* representation. The impostor selected for each object is that which most accurately represents it in the current viewing situation (some impostors for example are only convincing when viewed from certain angles). The result of the initial stage is that exactly one of the impostors of each object is marked as the available impostor of that object. It is this impostor that will be used to explicitly represent that object, if available cost allows.

The second stage of the algorithm is a greedy selection stage in which the selected representation of the entire scene is iteratively improved as much as the available cost will allow. The scene

representation selected at each stage is some subset of the set of available impostors of all the nested scene objects. Initially the selected representation consists of only the available impostor of the root (or scene) object. In each step the selected representation is improved by the replacement, if it can be afforded, of one selected available impostor by the available impostors of the children of that object. The object whose selected available impostor is replaced is that whose importance (as predicted by the *importance* heuristic) is greatest. In other words, the greedy selection favours more detailed representations of more important objects. The algorithm keeps track of the total cost of the selected representation (by means of incremental updates) and if an earmarked replacement cannot be afforded without exceeding the permitted maximum cost limit then that object is removed from consideration and its selected representation is scheduled for rendering. This greedy selection continues until no further replacements can be made. The scene representation selected for rendering then consists of those impostors scheduled for rendering during greedy selection.

The algorithm selects available representations for objects based on accuracy, without regard to cost, and then selects a scene representation from these available representations based on importance, without regard to value (perceptual benefit/cost) or cost. This constitutes a greedy selection based on importance (or, loosely, benefit).

Complexity

The time complexity of the algorithm of Maciel and Shirley is $O(n)$. The complexity of the initial stage, in which an available representation is chosen for each object, is $O(n)$ with respect to the number of impostors. Each object must be considered and there are $O(n)$ objects (on the assumption that the number of impostors belonging to each object is $O(1)$). The selection of the most appropriate representation for each object is $O(1)$ with respect to n. The complexity of the greedy stage, in which the selected representation of the scene is incrementally improved, is also $O(n)$ in the worst case.[9]

Optimality

No guarantee exists as to the optimality of the Maciel and Shirley greedy algorithm's solution to the hierarchical level of detail optimization problem. The optimality of the algorithms solution can

[9] Although in [47] it is claimed to be possible to reduce it to $O(\log n)$. We can see no way of doing this since in the worst case the selected representation of the scene must be exhaustively improved from the selected representation of the root object to the selected representations of the leaf objects. In this case all objects must be visited, so the complexity is $O(n)$.

be measured by the total profit of the algorithm's solution, where the profit of a selected impostor is defined by Maciel and Shirley to be the sum of the accuracy of the impostor and the importance of the object it represents [47]. In the worst case it may be arbitrarily bad. Firstly, the algorithm is capable of selecting available object representations that are very expensive and therefore represent poor value for their benefit. Secondly, we can imagine a situation in which the algorithm replaces a selected available impostor with the many expensive available impostors of its children, because one of them has a high importance.

Figure 15 shows a pathological instance of the hierarchical level of detail optimization problem for the algorithm of Maciel and Shirley. The problem illustrated by this example is that the algorithm selects objects for incrementation based on their importance, without regard to the accuracy or cost of their available representations. The algorithm selects the impostors of the left-most two leaf objects and the impostor of the right child of the root object, with a total profit of 3.4, while the optimal solution consists of the impostors of the right-most two leaf objects and the impostor of the left child of the root, with a total profit of 201.4. By varying the accuracies and costs of the impostors in the example the algorithm's solution may be made arbitrarily bad. We will return to the solution quality of the Maciel-Shirley algorithm in Section 4.3.

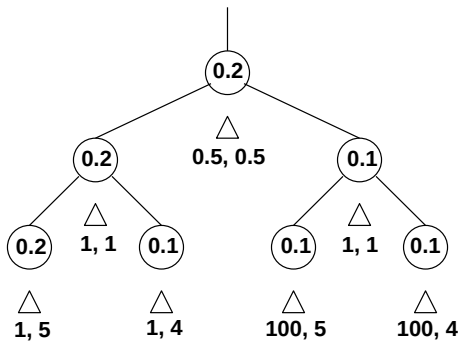


Figure 15: **Pathological example for the Maciel-Shirley algorithm.** The accuracy and cost of each available impostor is shown below the impostor, and the importance of each object is marked on that object. The rendering cost limit is 10.

2.7.2 Belblidia *et al* Algorithm

Belblidia *et al* [8] [7] present a predictive hierarchical level of detail optimization scheme that appears to be directly influenced by that of Maciel and Shirley. Like Maciel and Shirley, they use

a hierarchical level of detail description with shared object representations and a predictive level of detail optimization algorithm that limits the predicted rendering cost of the scene representation. Their approach is distinguished by the fact that they propose *two* algorithms for level of detail control: one that favours image quality while removing as much unnecessary detail as possible, and another that favours the regulation of frame rates and removes as much detail as is necessary to limit predicted rendering times. The first is essentially static and the other predictive. Their dual-algorithm approach is in contrast to the single algorithms of Funkhouser and Séquin and Maciel and Shirley, which attempt to maximize visual quality *while* regulating frame rates. The use of two separate algorithms emphasizes the fact that the approach taken in the predictive algorithm is not well-geared towards maximizing visual quality.

Like Maciel and Shirley, Belblidia *et al* use a greedy algorithm that incrementally replaces the selected impostors of objects with the available impostors of their children. However, unlike Maciel and Shirley, their greedy selection is based purely on the arbitrary order of the children rather than on any estimate of importance or representational accuracy. In our opinion this represents a step backwards from the already flawed greedy selection heuristic of Maciel and Shirley. Hence it cannot be expected to provide guaranteed or even approximately optimized levels of perceptual quality.

2.8 Summary

In this chapter we have discussed relevant background work, outlined more clearly the problem we intend to address, and discussed limitations of related previous work. We defined the level of detail optimization problem, being *the problem of dynamically and automatically selecting detail levels for each object in a visualization system so as to maximize the perceptual benefit of the selected scene representation, whilst ensuring that the rendering cost of the selected scene representation never exceeds some reasonable limit*.

We noted that the use of hierarchical level of detail descriptions in level of detail optimization is very common, and so defined the *hierarchical level of detail optimization problem*, being the level of detail optimization problem as it applies to hierarchical level of detail scene descriptions in which shared representations may be provided for groups of objects. However we noted that of the few predictive level of detail optimization algorithms proposed so far, the most promising is strictly non-hierarchical and does not permit the use of shared representations for groups of objects. For this reason we also defined the *non-hierarchical* level of detail optimization problem, being the predictive level of detail optimization problem for a strictly non-hierarchical level of detail

description in which objects are distinct and no shared representations are permitted for multiple objects.

We observed that the non-hierarchical level of detail optimization problem has been shown to be equivalent to the (NP-complete) Multiple Choice Knapsack Problem. We then showed that the hierarchical level of detail optimization problem was not.

Lastly we described several predictive level of detail optimization algorithms in detail. Two are non-hierarchical and the other two hierarchical. The non-hierarchical algorithm of Funkhouser and Séquin has a time complexity of $O(n \log n)$ in the worst case but is incremental and typically completes in only a few iterations. However its approach is flawed and it fails to provide solutions of guaranteed quality levels as is claimed. The other non-hierarchical algorithm, by Horvitz and Lengyel, is essentially a simplified version of the Funkhouser-Séquin approach. It provides quality guarantees but fails to cater effectively for multiple levels of detail. The hierarchical algorithm of Maciel and Shirley has a time complexity of $O(n)$ but is not incremental and performs a full optimization for every frame. Neither algorithm provides any guarantees as to the quality of its approximate solution, in spite of claims made by their authors. We aim in this thesis to examine more formally the hierarchical level of detail optimization problem and to devise new greedy optimization algorithms for it.

Chapter 3

Greedy Algorithm for the MCKP

In this chapter we present a greedy approximation algorithm for the Multiple Choice Knapsack Problem (MCKP), which was defined in Section 2.5. Recall from Section 2.6 that the MCKP was shown to be equivalent to the non-hierarchical level of detail optimization problem. This algorithm is therefore a first step towards an improved non-hierarchical level of detail optimization algorithm. In Chapters 4 and 6 we will look at extending the MCKP and this algorithm for it to cater for the more complex *hierarchical* level of detail optimization problem.

We actually present *two* greedy approximation algorithms for the MCKP in this Chapter. While the second is a complete algorithm that is guaranteed at least half-optimal for the entire problem, the first is a simplified algorithm that is half-optimal for a well-defined subproblem. We show that the simplified algorithm is likely to be useful in many practical applications, including level of detail optimization. This simplified algorithm has two advantages over the full algorithm: it has a lower time complexity, and it may be made *incremental* so that it accepts as input an initial best-guess solution derived from the previous problem instance. The ability to be made incremental makes the algorithm useful for level of detail optimization. It is this algorithm which we will extend to the hierarchical level of detail optimization problem in later chapters.

We begin in Section 3.1 by introducing a metric, *relative value*, that measures the value of items in the MCKP *relative to other items from the same candidate subset*. This metric represents the insight instrumental to the development of our algorithm: that when items in the MCKP are considered for selection by an approximation algorithm it is generally as *replacements* for previously selected items from the same candidate subset (recall from Definition 2.3 in Section 2.5.2 that only one item may be selected from each subset). It is this metric that allows us to formulate and prove correct approximation algorithms for the MCKP. In Section 3.2 we discuss the MCKP in general

and introduce the *convexity assumption*, a simplifying assumption about the nature of the MCKP that makes the first, simplified, greedy algorithm possible. In Section 3.3 we present the simplified algorithm. In Section 3.4 we prove that its solution is always at least half as good as the optimal solution, as long as the convexity assumption holds. In Section 3.5 we present the second, complete algorithm for the MCKP that does not depend on the convexity assumption. In Section 3.6 we prove that that algorithm's solution is always at least half-optimal. This proof is an extension of the proof presented for the simpler algorithm in Section 3.4.

In Section 3.7 we discuss the expected error of the algorithms *over and above* the worst case half-optimality guarantee and the implications of the convexity assumption. We show that the cases in which the assumption does not hold are relatively rare. In Section 3.8 we compare our algorithms with that of Funkhouser and Séquin (discussed in Section 2.6.1) and remark on the practical implications of the improvements that our algorithms represent. Recall from Chapter 2 that Funkhouser and Séquin [24] proposed a non-hierarchical level of detail optimization algorithm that was essentially a greedy approximation algorithm for the MCKP. We however showed that their algorithm suffers from several limitations, most notably that its solution is not guaranteed to be half-optimal as they claim. The algorithms presented here are therefore intended as corrections and replacements for theirs. In Section 3.9 we discuss the conversion of the simplified algorithm to an equivalent incremental version that exploit frame-to-frame coherence, with the aim of increasing efficiency in level of detail optimization. Finally in Section 3.10 we summarize the main points of the chapter.

3.1 Relative Value

In this section we define a metric, *relative value*, that is the core of both greedy algorithms for the MCKP described in this chapter. This metric was referred to as *slope* by Armstrong *et al* [5], where it was used in a greedy algorithm for C(MCKP), the continuous relaxation of MCKP.

Recall that in the greedy algorithm for the 0-1 KP described in Section 2.5.1, the *value* (profit / cost) of each item was used as a metric measuring the desirability of those items for selection. Items were simply ordered by descending value and selected if they could be afforded. The MCKP, by contrast, is characterized by the fact that exactly one item must be selected from every candidate subset. In the MCKP therefore the selection of an item is generally a *replacement* and entails not only the selection of that item at a certain cost and profit but also the *loss* of a previously selected item from the same candidate subset for a certain *decrease* in cost and profit. Alternatively we can regard the replacement of an item by another more expensive item as the selection of an imaginary

relative item that is the difference between the old item and the new (see Figure 16). Relative value essentially measures the value of this imaginary relative item.

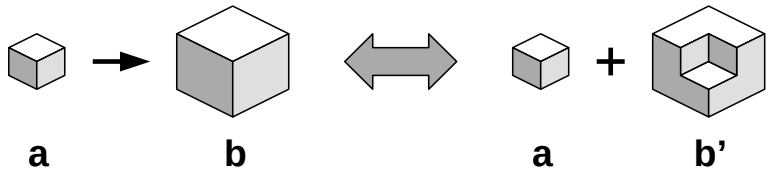


Figure 16: **Relative value.** When an item is selected in the MCKP, it is usually at the expense of some previously selected item from the same candidate subset. This figure shows how, when an item a is replaced by a more expensive item b, the effect is equivalent to the selection of an additional item b' that is the difference between a and b. The relative value of b *with respect to* a provides a measure of the value of b', the effect of replacing a with b.

Simply considering the *value* of each item without regard to the profit and cost of the item it replaces leaves the value-based greedy selection criterion of Funkhouser and Séquin (Section 2.5.4) open to discarding items that provide high profit for low cost in favour of items of slightly higher profit but with much greater cost in the MCKP.

In order to measure better the net effect of the selection of an item at the expense of a previously selected item from the same subset, we define a new metric, *relative value*, which measures the "desirability" of items as replacements for previously selected items from the same candidate subset:

Definition 3.1 *Relative Value*

The relative value *of a candidate item j with respect to another candidate item i from the same candidate subset is defined as:*

$$RV(j,i) = \frac{p_j - p_i}{w_j - w_i}$$

where p_j and w_j are the profit and cost of item j, and p_i and w_i are the profit and cost of item i respectively.

The relative value of an item is always measured relative to some other item from the same candidate subset. When we speak of the relative value of an item we will usually be referring to its relative value relative to the currently selected item from that subset.

3.2 Convexity Assumption

In this section we discuss the nature of the MCKP in general and introduce the *convexity assumption* upon which our simplified algorithm depends. It is useful to visualize the set of items in a candidate subset on a graph of increasing profit vs. increasing cost as shown in Figure 17. Each item is represented as a single point. It follows from the *IP-dominance*[1] of items with lower profit and greater cost by items from the same subset that the graph represented by the successive items must be monotonically increasing [49]. Any solution involving an IP-dominated item j can be trivially improved by replacing j with its dominating item i. Such dominated items may be removed as a preprocess to any MCKP algorithm [49].

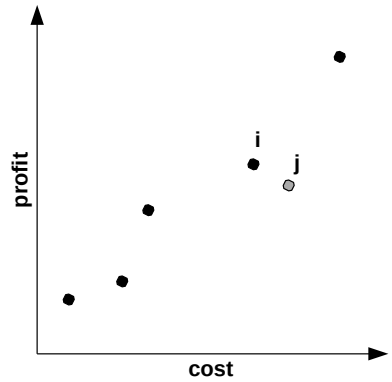


Figure 17: **A candidate subset represented on a profit vs. cost graph.** Each point represents an item, plotted on a graph of increasing profit vs. increasing cost. Due to the *IP-dominance* of items with lower profit and greater cost than other items from the same candidate subsets, the graph may be assumed to be strictly increasing. In this instance j is dominated by i and can be removed from consideration.

Our greedy algorithms for MCKP both select, as a conservative initial solution, the lowest cost item from each candidate subset. This ensures that every subset is represented. We then iteratively replace selected items with more expensive items from the same subset, as far as the available knapsack capacity will allow. Each replacement preserves the feasibility of the solution, since the replacing and replaced items always belong to the same candidate subset.

The items that are not selected initially are considered as replacements in a specific order which guarantees that items that provide greater "bang for the buck" are considered first. The *relative*

[1] IP stands for *Integer Programming*.

value metric is used to order the replacement items, since it provides a measure of the effect of replacing one item with another.[2] At each stage the relative value of each item under consideration is measured with respect to the currently selected item from its candidate subset, since this is the item which it may replace. The most desirable replacement item in each subset is that with *greatest relative value* with respect to the currently selected item from that subset. The most desirable replacement item from all subsets is that with greatest relative value with respect to its respective selected item. A simple example is provided in Figure 18.

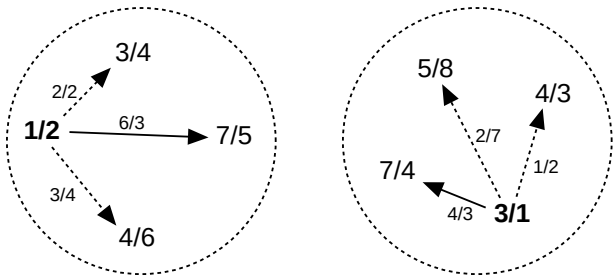


Figure 18: **Selection within candidate subsets.** In this example items are shown as *profit / cost* pairs. The currently selected item from each candidate subset is shown in bold, and available replacements are shown as arrows labeled with the relative values of the replacement items. The most desirable replacement in the first subset is $7/5$ with relative value $\frac{6}{3}$, and in the second subset it is $7/4$ with relative value $\frac{4}{3}$. The most desirable replacement from any subset is therefore $7/5$ as a replacement for $1/2$.

Since the relative value of each item is measured with respect to the currently selected item from its subset, whenever a selected item is replaced with another, the relative values of all the remaining items in that subset must be updated. This in general changes the ordering of those items in terms of relative value. In the case of our full MCKP algorithm (presented in Section 3.5), this is exactly what occurs. The situation is simplified significantly however if we assume that the items within each subset are ordered by descending relative value when they are ordered by ascending cost. We call this condition the *convexity assumption*:

Definition 3.2 *The convexity assumption*

The convexity assumption holds if, whenever there exist three items i, j, k in the same candidate

[2] Compare this with the use of *value* by Funkhouser and Séquin in their greedy algorithm described in Section 2.5.4.

subset such that $w_i \leq w_j \leq w_k$*, it is true that*

$$RV(j,i) \quad \geq \quad RV(k,i).$$

When the convexity assumption holds the graph of every candidate subset is convex, as shown in Figure 19. In that case our MCKP algorithm can get by with always considering only the items with immediately higher cost than the currently selected items in each candidate subset. Furthermore the ordering of items within each candidate subset in terms of descending relative value with respect to the currently selected item in each subset does not change as new items are selected. An algorithm exploiting the assumption need only consider the items in each candidate subset in *ascending order of cost*:at each stage the most desirable unconsidered replacement item in each subset is that with lowest cost. Our simplified MCKP algorithm, which we present in Section 3.3, is based on this assumption.

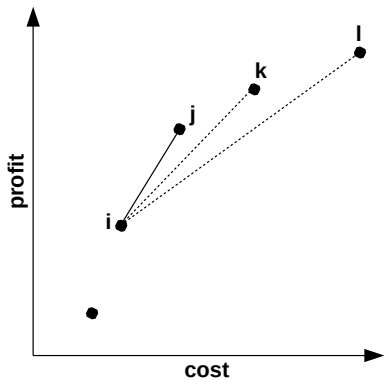


Figure 19: **A convex candidate subset.** When the *convexity assumption* holds, the items in each candidate subset decrease in relative value as they increase in cost. The relative value of an item with respect to another corresponds to the gradient of the line between them.

3.3 Simplified Algorithm

In this section we describe our simplified greedy algorithm for the MCKP that makes use of the relative value metric described in Section 3.1 and exploits the *convexity assumption* introduced in Section 3.2.

input: an instance of the MCKP (see definition 2.3)
output: a feasible solution to that instance

begin
set $s \leftarrow 0, L \leftarrow$ empty list // no critical item yet, empty list
set $B \leftarrow \{b_1, b_2, b_3, \ldots, b_r\}$ where b_k is the cheapest item in subset N_k
for $k \leftarrow 1, 2, 3, \ldots, r$ // for each candidate subset
{
order the items in subset N_k by ascending cost
set $x_{b_k} \leftarrow 1$ // select the cheapest item in subset N_k
for each item $l \in N_k, l \neq b_k$ // for all other items in subset
{
set $m \leftarrow$ the immediately lower cost item from N_k than l
set $x_l \leftarrow 0$, relative value$_l \leftarrow \frac{p_l - p_m}{w_l - w_m}$
insert l into L // add l to the list of unconsidered items
}
}
order the items in L by descending relative value
for each item j in L // for each unconsidered item
{
set $l \leftarrow$ the item from the same candidate subset N_k as j such that $x_l = 1$
if $\sum_{i \in N} x_i w_i - w_l + w_j \leq c$ **then** // if we can afford to replace l with j
set $x_l \leftarrow 0, x_j \leftarrow 1$ // replace l with j
else
if $s = 0$ **then set** $s \leftarrow j$ // if there is no critical item yet, j is critical
}
if $s \neq 0$ **then** // if there is a critical item
{ // then consider the critical item solution
set $t \leftarrow$ the item in B from the same candidate subset as s
if $\sum_{i \in B} p_i - p_t + p_s > \sum_{i \in N} x_i p_i$ **and** $\sum_{i \in B} w_i - w_t + w_s \leq c$ **then**
set $x_i \leftarrow \begin{cases} 1 & \text{if } i = s \text{ or } (i \in B \text{ and } i \neq t) \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in N$
}
end

Figure 20: **The simplified greedy algorithm for the MCKP.**

The algorithm is shown in Figure 20, and its operation is illustrated by an example in Figure 21. The algorithm begins by inserting into the knapsack a minimal initial solution consisting of the lowest cost item from each candidate subset. This initial solution is then iteratively improved by the successive replacement of selected items with the immediately more expensive items from their candidate subsets, as far as the available space in the knapsack will allow. The unconsidered items at any stage are considered for selection in *descending order of relative value* (with respect to the currently selected items from their respective subsets). By virtue of the convexity assumption (Section 3.2), this also constitutes an ordering within subsets by increasing cost. Therefore only the lowest cost item in each subset must be considered at each stage. An item under consideration replaces the item from the same subset already in the knapsack if the replacement can be afforded; otherwise it is discarded.

After the greedy selection phase the resulting solution is compared to the *critical item solution*. The *critical item* is the first item to be rejected during greedy selection due to its cost being too high.[3] The critical item solution consists of the critical item and the lowest cost items from each of the other candidate subsets. If the critical item solution has greater profit and can be afforded then it is selected instead. The critical item check serves the same purpose as in the greedy algorithm for 0-1 KP (See Section 2.5.3). It guards against the pathological case in which the particular sizes of the knapsack and items favours the selection of a solution of low value that happens to fit the available space but whose selection would otherwise be prevented by the selection of higher valued items of low profit.

The time complexity of the simplified algorithm is $O(n \log n)$ in the number of candidate items, due to the sorting step in which the items are ordered by descending relative value. The critical item is found automatically as a by-product of the greedy selection stage. An immediate optimization that increases the efficiency of the algorithm is to discount all remaining items within a candidate subset as soon as any item from that subset has been found to be too expensive. Since the items in each subset are considered in ascending order of cost, any remaining items must also be too expensive.

[3] Not counting items that are too expensive to be elements of any feasible solutions — if such *infeasible items* exist then they should be discarded as a preprocess.

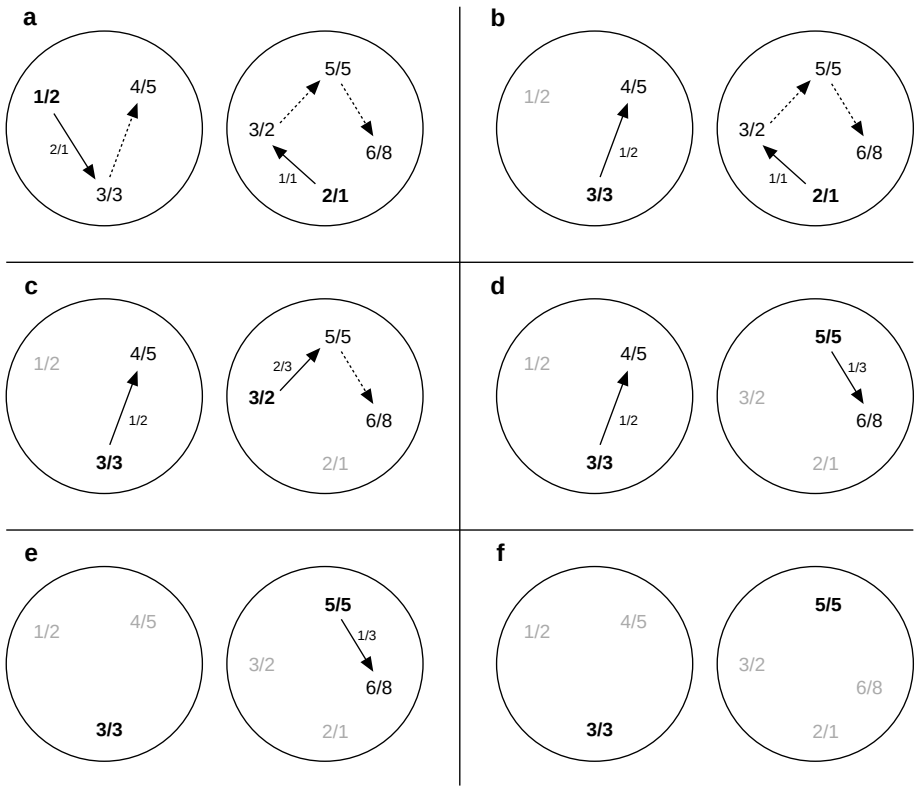


Figure 21: **Example execution of the simplified MCKP algorithm.** This example shows the operation of the simplified MCKP algorithm for an instance of the MCKP with 7 items (shown as profit/cost pairs) partitioned into 2 candidate subsets (circles), and with a knapsack size of 9. (a) Initially the items in each subset are ordered by ascending cost (indicated by the arrows) and the cheapest item in each subset is selected (bold items). Arrows leading to items available for selection are labeled with the relative values of those items. The available replacement item with greatest relative value is 3/3 in the first candidate subset. It can be afforded and so is selected, replacing 1/2. (b) 3/2 has greatest relative value and can be afforded, so it replaces 2/1. (c) 5/5 has greatest relative value and can be afforded, so it replaces 3/2. (d) 4/5 has greatest relative value but cannot be afforded, so it is discarded. (e) 6/8 is the only remaining item but cannot be afforded, so it is discarded. (f) The algorithm's solution is 3/3 and 5/5, for a total profit of 8. The critical item is 4/5, and the critical item solution is 4/5 and 2/1, for a total profit of 6.

3.4 Proof of Half-Optimality for the Simplified Algorithm

In this section we prove the half-optimality of the MCKP greedy algorithm described in Section 3.3, for the subproblem of the MCKP defined by the *convexity assumption* introduced in Section 3.2. Recall that the convexity assumption holds when the relative values of items within candidate subsets always decrease monotonically with increasing cost. The proof is an extension to the MCKP of the general approach embodied by the proof for the 0-1 KP algorithm presented in Section 2.5.1.

3.4.1 Overview of Proof

The proof consists of six steps:

1. We formulate an equation relating the profit of the optimal solution to that of the solution G reached at an intermediate stage in the greedy algorithm immediately before the consideration and rejection of the critical item. This equation provides an upper bound on the maximum error of the greedy algorithm and includes terms that quantify the profit lost by not selecting items that are in the optimal solution and the profit gained by selecting items that are not.

2. We show that any items that are in the optimal solution but were not selected by the greedy algorithm before the rejection of the critical item must have lower relative value than the critical item, since they were not selected before it.

3. Similarly we show that any items that were selected by the greedy algorithm before the rejection of the critical item but are not in the optimal solution must at least have higher relative value than the critical item, since they were selected before it.

4. Using the results of steps 2 and 3, we show that the maximum error of the greedy algorithm is bounded by the total difference in cost between the optimal solution and G, multiplied by the relative value of the critical item.

5. Then we show that the difference in cost between the optimal solution and G is bounded by the difference in cost between the critical item and the item from the same candidate subset that is selected in the intermediate greedy solution.

6. Substituting, we show that the maximum error of G is bounded by the difference in profit between the critical item and the selected item from the same subset. Recalling that the algorithm also considers the critical item solution, we conclude that the algorithm's solution is at least half as good as the optimal one.

3.4.2 Proof

1. Given an instance of the MCKP, let the profit of the optimal solution to this instance be z. Let G be the set of items in the intermediate solution reached by the greedy algorithm immediately before the critical item is considered (and rejected), and let $z^g = \sum_{i \in G} p_i$ be the profit of this partial greedy solution.[4]

   Recall that both the optimal solution and the intermediate greedy solution G must contain exactly one item from every candidate subset. Therefore the profit of the optimal solution is equal to the profit z^g of the partial greedy solution G plus, for each candidate subset N_k, the difference between the profits of the item $o_k \in N_k$ which is in the optimal solution and the item $g_k \in N_k$ selected in G:

   $$z = z^g + \sum_{k=1}^{r} (p_{o_k} - p_{g_k}).$$

   Therefore

   $$z = z^g + \sum_{k \in A} (p_{o_k} - p_{g_k}) - \sum_{k \in B} (p_{g_k} - p_{o_k}) \tag{14}$$

   where A is the set of candidate subsets for which o_k has greater cost (and profit) than g_k, and B is the set of candidate subsets for which g_k has greater cost (and profit) than o_k. In other words, A contains those candidate subsets in which the algorithm selected "less" than it should have, and B contains those where it selected "more" than it should have.

2. In this step we consider the candidate subsets N_k that are in A. Recall that these are those for which the item $o_k \in N_k$ in the optimal solution has greater cost than the item $g_k \in N_k$ selected in G. We show that in such cases the item o_k in the optimal solution must have lower relative value (with respect to g_k) than the critical item s (with respect to the item t from the same subset as s, that is selected in G). This follows from the fact that s was chosen for selection and o_k was not.

   When the critical item s was chosen for selection (and rejected) the set of currently selected items was exactly G. Therefore the critical item was considered as a replacement for some item t that is an element of G, and was considered instead of some item v_k that is the element of N_k of immediately higher cost than g_k. Therefore v_k has lower relative value (with respect

[4] In practice the algorithm may go on to select other later items, replacing items in G, but since every replacement increases the total profit of the selected items, we know that the profit of the final greedy solution is greater than or equal to z^g.

to g_k) than s (with respect to t):

$$\frac{p_{v_k} - p_{g_k}}{w_{v_k} - w_{g_k}} \leq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad k \in A.$$

Now because the items in each candidate subset decrease in relative value as they increase in cost (from the convexity assumption — Definition 3.2), it must be true that

$$\frac{p_{o_k} - p_{g_k}}{w_{o_k} - w_{g_k}} \leq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad k \in A. \tag{15}$$

3. In this step we consider the candidate subsets N_k that are in B. Recall that these are those for which the item $o_k \in N_k$ in the optimal solution has *lower* cost than the item $g_k \in N_k$ selected in G. We show that in such cases the item g_k in G must have greater relative value (with respect to o_k) than the critical item s (with respect to t). This follows from the fact that g_k was chosen for selection before s.

   When the critical item s was chosen for selection (and rejected) the set of currently selected items was exactly G. There must therefore exist a sequence of items $j_1, j_2, j_3, \ldots, j_z$ in N_k from o_k to g_k (that is, $j_1 = o_k$ and $j_z = g_k$) such that j_{i+1} is selected as the replacement for j_i for all of $i = 1, 2, 3, \ldots, z-1$.

   Likewise there must exist a sequence of items $m_1, m_2, m_3, \ldots, m_y$ in the candidate subset containing s and t where m_1 is the cheapest item in that candidate subset, $m_y = s$, $m_{y-1} = t$, and m_{i+1} is selected by the algorithm as the replacement for m_i for all of $i = 1, 2, 3, \ldots, y-2$. Note that m_y is not selected as the replacement for m_{y-1}, because m_y (ie. s) is not selected at all.

   Then we know that the algorithm at some stage replaced j_{z-1} with j_z (ie. g_k) instead of replacing some item m_u in the sequence $m_1, m_2, m_3, \ldots, m_{y-1}$ with m_{u+1}, since j_z was selected and s was not. Therefore

   $$\frac{p_{g_k} - p_{j_{z-1}}}{w_{g_k} - w_{j_{z-1}}} \geq \frac{p_{m_{u+1}} - p_{m_u}}{w_{m_{u+1}} - w_{m_u}}.$$

   Now because the items in each candidate subset are considered in order of ascending cost and therefore descending relative value (from the convexity assumption — Definition 3.2) it must be true that

   $$\frac{p_{g_k} - p_{o_k}}{w_{g_k} - w_{o_k}} \geq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad k \in B. \tag{16}$$

4. Therefore, from (14), (15) and (16) we have

$$z \leq z^g + \sum_{k \in A}(w_{o_k} - w_{g_k})\frac{p_s - p_t}{w_s - w_t} - \sum_{k \in B}(w_{g_k} - w_{o_k})\frac{p_s - p_t}{w_s - w_t} \quad (17)$$

$$\leq z^g + \left[\sum_{k \in A}(w_{o_k} - w_{g_k}) - \sum_{k \in B}(w_{g_k} - w_{o_k})\right]\frac{p_s - p_t}{w_s - w_t} \quad (18)$$

$$\leq z^g + \sum_{k=1}^{r}(w_{o_k} - w_{g_k})\frac{p_s - p_t}{w_s - w_t}. \quad (19)$$

5. Let $\overline{c} = c - \sum_{i \in G} w_i$ be the space left in the knapsack after the selection of G, immediately before the rejection of the critical item s. From the fact that s was rejected we know that the difference in cost between s and t is greater than $\overline{c}$:

$$\overline{c} < w_s - w_t. \quad (20)$$

Furthermore we know that the total difference in cost between the items in the optimal solution and the items from the same candidate subsets in the greedy solution must be less than or equal to $\overline{c}$:

$$\sum_{k=1}^{r}(w_{o_k} - w_{g_k}) \leq \overline{c}.$$

Therefore, from (20),

$$\sum_{k=1}^{r}(w_{o_k} - w_{g_k}) < w_s - w_t. \quad (21)$$

6. Therefore, from (19) and (21),

$$z \leq z^g + (w_s - w_t)\frac{p_s - p_t}{w_s - w_t} \quad (22)$$

$$\leq z^g + p_s - p_t \quad (23)$$

$$\leq z^g + p_s. \quad (24)$$

Recall that the greedy algorithm compares the total profit of the final greedy solution (which is greater than or equal to z^g) to the total profit z^s of the cheapest solution containing the critical item (which is clearly greater than p_s), and keeps whichever solution is better. That is, the algorithm's solution has profit $z^h \geq \max(z^g, p_s)$. Therefore, from (24),

$$z^h \geq \frac{1}{2}z$$

and the profit of the algorithm's solution is guaranteed to be at least half the profit of the optimal solution.

The critical item check is complicated slightly by the fact that the critical item by itself is not a feasible solution; hence the algorithm constructs a complete solution containing the cheapest item from every other subset. However it is certain that this lowest cost critical item solution will be feasible, since otherwise the critical item would not be a feasible item itself and would have been removed in the reduction pre-process.

3.5 Full Algorithm

In this section we present our full algorithm for the MCKP, which we prove (in Section 3.6) is at least half-optimal for the entire problem. The algorithm differs from the simplified one presented in Section 3.3 in that it does not make use of the convexity assumption. The selection of items in descending order of relative value may therefore not constitute an ordering by ascending cost within candidate subsets. The algorithm must therefore be prepared to consider, at each stage, any of the remaining items in each candidate subset. When the most desirable replacement item is too expensive to be selected, the algorithm must also consider other "concave" items from the same subset with lower relative value.

The algorithm is shown in Figure 22. Its operation is illustrated by an example in Figure 23. Like the simplified algorithm it begins by selecting as an initial solution the lowest cost item from each candidate subset. The remaining items are considered one by one as potential replacements for currently selected items. Each replacement increases the total cost and profit of the solution. The potential replacement items are considered in descending order of relative value, always measured with respect to the currently selected items from their respective subsets. At each stage the item with greatest relative value with respect to its respective selected item is considered for selection. If it can be afforded, it is selected and the currently selected item from that subset is discarded. The relative values of the remaining items from that subset are updated (with respect to the newly selected item) and any unconsidered items with lower cost than the newly selected item are discarded. Since these discarded items have lower costs than the selected item they are assumed, essentially, to have been briefly selected and immediately replaced by the more expensive newly selected item.

In the event that the potential replacement cannot be afforded the item under consideration is simply discarded and is marked as the *critical item* s if it is the first item to be rejected in this way. Like the simplified algorithm of Section 3.3, the full algorithm considers the *critical item solution* consisting of the critical item and the lowest cost items from all other subsets.

The algorithm assumes that there are no items that have greater cost but lower profit than other

input: an instance of the MCKP (see definition 2.3)
output: a half-optimal or better solution to that instance

begin
  set $s \leftarrow 0, L \leftarrow$ empty list // no critical item yet, empty list
  set $B \leftarrow \{b_1, b_2, b_3, \ldots, b_r\}$ where b_k is the cheapest item in subset N_k
  for $k \leftarrow 1, 2, 3, \ldots, r$ // for each candidate subset
  {
    set $x_{b_k} \leftarrow 1$ // select the cheapest item in subset N_k
    for each item $l \in N_k, l \neq b_k$ // for all other items in subset
      set $x_l \leftarrow 0$, relative value$_l \leftarrow \mathrm{RV}(l, k), L \leftarrow L \cup \{l\}$
  }
  while L is not empty // while there are unconsidered items
  {
    set $j \leftarrow$ the item with greatest relative value in L
    set $l \leftarrow$ the item from the same candidate subset N_k as j such that $x_l = 1$
    set $L \leftarrow L - \{j\}$ // remove j from L
    if $\sum_{i \in N} x_i w_i - w_l + w_j \leq c$ **then** // if we can afford to replace l with j
    {
      set $x_l \leftarrow 0, x_j \leftarrow 1$ // replace l with j
      for each item $m \in N_k$ in L // update the candidate subset containing j
        if $w_m \leq w_j$ **then** // if m has lower cost than j
          set $L \leftarrow L - \{m\}$ // discard m
        else // else calculate new relative value of m
          set relative value$_m \leftarrow \mathrm{RV}(m, j)$
    }
    else if $s = 0$ **then set** $s \leftarrow j$ // if there is no critical item, j is critical
  }
  if $s \neq 0$ **then** // if there is a critical item
  { // then consider the critical item solution
    set $t \leftarrow$ the item in B from the same candidate subset as s
    if $\sum_{i \in B} p_i - p_t + p_s > \sum_{i \in N} x_i p_i$ **and** $\sum_{i \in B} w_i - w_t + w_s \leq c$ **then**
      set $x_i \leftarrow \begin{cases} 1 & \text{if } i = s \text{ or } (i \in B \text{ and } i \neq t) \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in N$
  }
end

Figure 22: **Full greedy algorithm for the MCKP.**

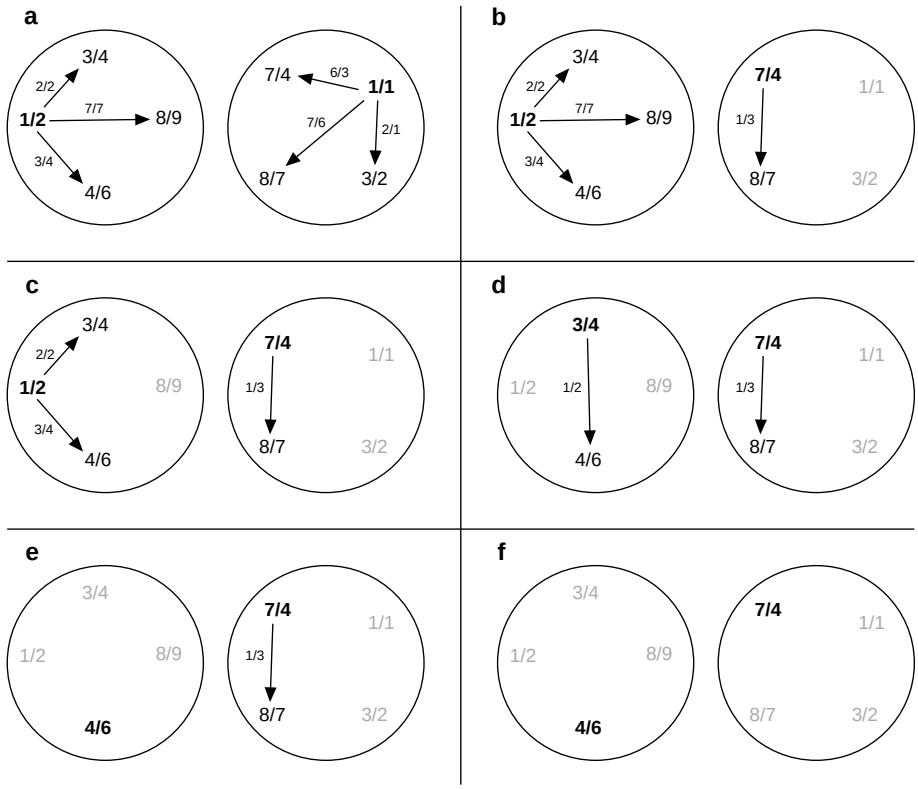


Figure 23: **Example execution of the full MCKP algorithm.** This example shows the operation of the full MCKP algorithm for an instance of the MCKP with 8 items (shown as profit/cost pairs) partitioned into 2 candidate subsets (circles), and with a knapsack size of 10. (a) Initially the lowest cost item from each subset is selected (bold items). Arrows lead to the available replacement items, labeled with the relative values of those items with respect to the currently selected item from that subset. The greatest relative value, 2.0, is shared by two replacement items: 7/4 and 3/2. The algorithm arbitrarily chooses 7/4, replacing 1/1 with 7/4 and discarding 3/2 since it has lower cost than 7/4. The relative value of the remaining item, 8/7, is updated. (b) 8/9 has greatest relative value but cannot be afforded, so it is discarded. (c) 3/4 has greatest relative value and can be afforded, so it replaces 1/2. (d) 4/6 has greatest relative value and can be afforded, so it replaces 3/4. (e) 8/7 is the only remaining item but cannot be afforded, so it is discarded. (f) The algorithm's solution is 4/6 and 7/4, for a total profit of 11. The critical item is 8/9, and the critical item solution is 8/9 and 1/1, with a total profit of 9.

items from the same candidate subsets. These items are *IP-dominated* (see Section 3.2) and must be removed in a reduction preprocess phase [49]. The algorithm also assumes that no *infeasible items* exist that are not elements of any feasible solution (because their cost is too high to allow them to be selected with any combination of other items). These *infeasible* items can also be removed during the reduction phase.

The complexity of the algorithm is $O(n(r+m))$, where r is the number of candidate subsets and m is the number of items in the largest subset. In an efficient implementation of the algorithm, each candidate subset may be stored as a separate list. If the subset lists are stored in arbitrary unsorted order then a pointer is associated with each one marking the item in that subset with greatest relative value. After the selection of an item, redundant items are removed by traversing the list and deleting any items with lower cost than the newly selected item. In the same traversal, the relative values of the remaining items are recalculated. After the update traversal the index is updated to point to the item with highest relative value. The complexity of the entire update step is $O(m)$, where m is the size of the largest candidate subset. The search for the item with greatest relative value from any subset involves traversing the list of the best items from each subset, pointed to by the indices associated with the subsets. The complexity of this is $O(r)$, where r is the number of subsets. Each candidate item must be considered exactly once. Therefore the complexity of the entire algorithm is $O(n(r+m))$.

Note that the *average* size of the candidate subsets is inversely proportional to the number of candidate subsets r. In the worst case either r or m may be equal to n, in which case the other is obviously equal to 1. In this worst case the complexity of the algorithm is $O(n^2)$. Intuitively, the selection of each of the n items involves the comparison of up to n candidate subsets (in the case where $r = n$) or up to n items within one candidate subset (in the case where $m = n$). We note however that in this worst case MCKP actually degenerates into simpler problems: 0-1 KP, in the case when $m = 1$, and the selection of a single item from a set of candidates in the case where $r = 1$. In typical practical problems neither m or r is close to n, so that the average complexity is $O(n(r+m))$.

The complexity of the reduction phase is $O(n \log m)$ [49]. It may be incorporated into the initialization of the greedy algorithm itself.

3.6 Proof of Half-Optimality for the Full Algorithm

In Section 3.5 we presented a second and more complex greedy algorithm for the MCKP. In this section we prove that that algorithm's solution to the MCKP is always at least half as good as the optimal solution. Unlike the proof for the simplified algorithm presented in Section 3.4, this proof does not depend on the *convexity assumption* introduced in Section 3.2. The proof is an extension of the earlier proof, and follows the same six-step general plan (see Section 3.4). While steps 1, 4, 5 and 6 are identical to those in Section 3.4, steps 2 and 3 are complicated by the added complexity of the second algorithm.

A key feature of the simplified algorithm is that the items in each candidate subset are always considered in ascending order of profit and cost and descending order of relative value. In the case of the full algorithm this is no longer true, since items are sometimes selected out of ascending cost order within the same subset. When this occurs, any items with lower cost from that subset are simply discarded (see Section 3.5). We say that these items are *implicitly selected* and immediately replaced by the more expensive *explicitly selected* items. The following Lemma shows that the items in each subset that are *explicitly* selected are selected in descending order of relative value:

Lemma 3.1 *If $m_1, m_2, m_3, \ldots, m_y$ are elements of the same candidate subset N_k such that m_{i+1} is selected explicitly by the algorithm as the replacement for m_i for all of $i = 1, 2, 3, \ldots, y-1$ then the relative value of m_{i+1} with respect to m_i is greater than the relative value of m_{i+2} with respect to m_{i+1}:*

$$\frac{p_{m_{i+1}} - p_{m_i}}{w_{m_{i+1}} - w_{m_i}} \geq \frac{p_{m_{i+2}} - p_{m_{i+1}}}{w_{m_{i+2}} - w_{m_{i+1}}} \qquad i = 1, 2, 3, \ldots, y-2.$$

Proof:

For any three consecutive items m_i, m_{i+1}, m_{i+2} in $m_1, m_2, m_3, \ldots, m_y$ we know that $w_{m_i} \leq w_{m_{i+1}} \leq w_{m_{i+2}}$ (by definition of the algorithm) and that $p_{m_i} \leq p_{m_{i+1}} \leq p_{m_{i+2}}$ (because of the dominance of items with lower profit and greater cost by other items from the same candidate subset). Furthermore we know that m_{i+1} was selected as the replacement for m_i and m_{i+2} was not, so

$$\frac{p_{m_{i+1}} - p_{m_i}}{w_{m_{i+1}} - w_{m_i}} \geq \frac{p_{m_{i+2}} - p_{m_i}}{w_{m_{i+2}} - w_{m_i}}.$$

Therefore, by considering the triangle formed by m_i, m_{i+1}, m_{i+2} (see Figure 24) we deduce that

$$\frac{p_{m_{i+2}} - p_{m_{i+1}}}{w_{m_{i+2}} - w_{m_{i+1}}} \leq \frac{p_{m_{i+1}} - p_{m_i}}{w_{m_{i+1}} - w_{m_i}}.$$

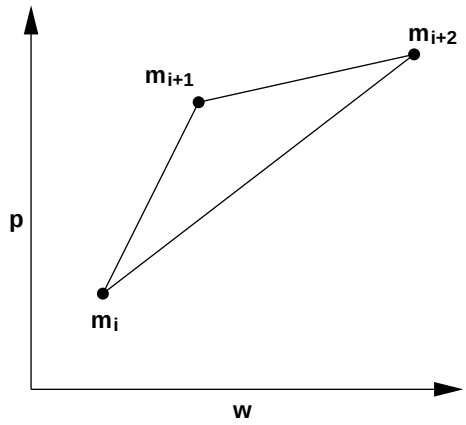


Figure 24: **Triangle formed by m_i, m_{i+1} and m_{i+2}.** Given that the relative value of m_{i+1} with respect to m_i is greater than or equal to the relative value of m_{i+2} with respect to m_i (and knowing that $w_{m_i} \leq w_{m_{i+1}} \leq w_{m_{i+2}}$ and $p_{m_i} \leq p_{m_{i+1}} \leq p_{m_{i+2}}$) we deduce that the relative value of m_{i+2} with respect to m_{i+1} is less than or equal to the relative value of m_{i+2} with respect to m_i. The slopes of the lines in the graph represent visually the relative values of items with respect to other items. For example, the relative value of m_{i+2} with respect to m_i is shown by the slope of the line from m_i to m_{i+2}.

3.6.1 Proof

1. Given an instance of the MCKP, let the profit of the optimal solution to this instance be z. Let G be the set of items in the intermediate solution reached by the greedy algorithm immediately before the critical item is considered (and rejected), and let $z^g = \sum_{i \in G} p_i$ be the profit of this partial greedy solution.

   Recall that both the optimal solution and the intermediate greedy solution G must contain exactly one item from every candidate subset. Therefore the profit of the optimal solution is equal to the profit z^g of the partial greedy solution G plus, for each candidate subset N_k, the difference between the profits of the item $o_k \in N_k$ which is in the optimal solution and the item $g_k \in N_k$ selected in G:

   $$z = z^g + \sum_{k=1}^{r} (p_{o_k} - p_{g_k}).$$

   Therefore

   $$z = z^g + \sum_{k \in A} (p_{o_k} - p_{g_k}) - \sum_{k \in B} (p_{g_k} - p_{o_k}) \tag{25}$$

where A is the set of candidate subsets for which o_k has greater cost (and profit) than g_k, and B is the set of candidate subsets for which g_k has greater cost (and profit) than o_k. In other words, A contains those candidate subsets in which the algorithm selected "less" than it should have, and B contains those where it selected "more" than it should have.

2. In this step we consider the candidate subsets N_k that are in A. Recall that these are those for which the item $o_k \in N_k$ in the optimal solution has greater cost than the item $g_k \in N_k$ selected in G. As in the simplified proof, we show that in such cases the item o_k in the optimal solution must have lower relative value (with respect to g_k) than the critical item s (with respect to the item t from the same subset as s, that is selected in G). This follows from the fact that s was chosen for selection and o_k was not.

   When the critical item s was chosen for selection (and rejected) the set of currently selected items was exactly G. Therefore the critical item was considered as a replacement for some item t that is an element of G, and was chosen instead of selecting o_k as a replacement for g_k (which would have been possible, since the full algorithm considers not only the items of immediately higher cost). Therefore o_k has lower relative value with respect to g_k than s does with respect to t:

$$\frac{p_{o_k} - p_{g_k}}{w_{o_k} - w_{g_k}} \leq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad k \in A. \tag{26}$$

3. In this step we consider the candidate subsets N_k that are in B. Recall that these are those for which the item $o_k \in N_k$ in the optimal solution has *lower* cost than the item $g_k \in N_k$ selected in G. As in the simplified proof, we show that in such cases the item g_k in G must have greater relative value (with respect to o_k) than the critical item s (with respect to t). This follows from the fact that g_k was chosen for selection before s.

   When the critical item s was chosen for selection (and rejected) the set of currently selected items was exactly G. Therefore there must exist a sequence $j_1, j_2, j_3, \ldots, j_z$ of elements of N_k from o_k to g_k such that j_{i+1} is explicitly selected as the replacement for j_i for all of $i = 1, 2, 3, \ldots, z-1$. This step is complicated by the fact that, in the full MCKP algorithm, some items are never explicitly selected (but only implicitly selected). We know that g_k is explicitly selected (since $g_k \in G$), but o_k may have been *implicitly* selected by the selection of another element of N_k of higher cost than o_k. Therefore let j_1 be the highest cost explicitly selected element of N_k such that $w_{j_1} \leq w_{o_k}$[5], and let $j_z = g_k$. Note that $w_{j_2} \geq w_{o_k}$.

[5] In the case where o_k is explicitly selected, j_1 is o_k itself.

There must also exist a sequence of items $m_1, m_2, m_3, \ldots, m_y$ in the candidate subset containing s and t leading from the selection of the cheapest item in that subset to the selection of t, where m_1 is the cheapest item in the candidate subset, $m_y = s$, $m_{y-1} = t$, and m_{i+1} is selected explicitly by the algorithm as the replacement for m_i for all of $i = 1, 2, 3, \ldots, y-2$. Note that m_y is not selected as the replacement for m_{y-1}, because m_y (ie. s) is not selected at all.

Since s was chosen for selection *after* g_k, for every j_i in the sequence $j_1, j_2, j_3, \ldots, j_{z-1}$ there must exist an item m_u in the sequence $m_1, m_2, m_3, \ldots, m_{y-1}$ such that

$$\frac{p_{j_{i+1}} - p_{j_i}}{w_{j_{i+1}} - w_{j_i}} \geq \frac{p_s - p_{m_u}}{w_s - w_{m_u}}. \tag{27}$$

From Lemma 3.1 we know that the items in $m_1, m_2, m_3, \ldots, m_y$ are in descending order of relative value, and therefore that

$$\frac{p_{m_{i+1}} - p_{m_i}}{w_{m_{i+1}} - w_{m_i}} \geq \frac{p_{m_y} - p_{m_{y-1}}}{w_{m_y} - w_{m_{y-1}}} \quad \forall \quad i \in \{u, \ldots, y-1\}.$$

Rewriting,

$$\frac{p_{m_{i+1}} - p_{m_i}}{w_{m_{i+1}} - w_{m_i}} \geq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad i \in \{u, \ldots, y-1\}.$$

Therefore, since

$$\frac{p_s - p_{m_u}}{w_s - w_{m_u}} = \frac{\sum_{i=u}^{y-1}(p_{m_{i+1}} - p_{m_i})}{\sum_{i=u}^{y-1}(w_{m_{i+1}} - w_{m_i})},$$

we know that

$$\frac{p_s - p_{m_u}}{w_s - w_{m_u}} \geq \frac{p_s - p_t}{w_s - w_t}.$$

Therefore, from (27),

$$\frac{p_{j_{i+1}} - p_{j_i}}{w_{j_{i+1}} - w_{j_i}} \geq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad i \in \{1, \ldots, z-1\}.$$

Then since

$$\frac{p_{j_z} - p_{j_1}}{w_{j_z} - w_{j_1}} = \frac{\sum_{i=1}^{z-1}(p_{j_{i+1}} - p_{j_i})}{\sum_{i=1}^{z-1}(w_{j_{i+1}} - w_{j_i})}$$

we know that

$$\frac{p_{j_z} - p_{j_1}}{w_{j_z} - w_{j_1}} \geq \frac{p_s - p_t}{w_s - w_t}.$$

Substituting g_k with j_z, we have shown that the relative value of g_k with respect to j_1 is at least as high as the relative value of s with respect to t:

$$\frac{p_{g_k} - p_{j_1}}{w_{g_k} - w_{j_1}} \quad \geq \quad \frac{p_s - p_t}{w_s - w_t}. \tag{28}$$

Now if o_k is explicitly selected (that is, if $j_1 = o_k$) then by substitution into (28) it follows that the relative value of g_k with respect to o_k is at least as high as the relative value of s with respect to t (which is what we wanted to show in this step):

$$\frac{p_{g_k} - p_{o_k}}{w_{g_k} - w_{o_k}} \geq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad k \in B; \tag{29}$$

Otherwise if o_k is selected implicitly by the explicit selection of j_2 then $j_1 \neq o_k$ and

$$w_{j_1} \leq w_{o_k} \leq w_{j_2}.$$

Then, since the algorithm selected j_2 over o_k as a replacement for j_1,

$$\frac{p_{j_2} - p_{j_1}}{w_{j_2} - w_{j_1}} \quad \geq \quad \frac{p_{o_k} - p_{j_1}}{w_{o_k} - w_{j_1}}.$$

Therefore, by considering the triangle formed by j_1, j_2 and o_k (Figure 25) we deduce that the relative value of j_2 with respect to o_k is at least as high as the relative value of j_2 with respect to j_1:

$$\frac{p_{j_2} - p_{o_k}}{w_{j_2} - w_{o_k}} \geq \frac{p_{j_2} - p_{j_1}}{w_{j_2} - w_{j_1}}.$$

Then, since

$$\frac{p_{g_k} - p_{o_k}}{w_{g_k} - w_{o_k}} = \frac{(p_{g_k} - p_{j_2}) + (p_{j_2} - p_{o_k})}{(w_{g_k} - w_{j_2}) + (w_{j_2} - w_{o_k})}$$

and

$$\frac{p_{g_k} - p_{j_1}}{w_{g_k} - w_{j_1}} = \frac{(p_{g_k} - p_{j_2}) + (p_{j_2} - p_{j_1})}{(w_{g_k} - w_{j_2}) + (w_{j_2} - w_{j_1})}$$

we know that

$$\frac{p_{g_k} - p_{o_k}}{w_{g_k} - w_{o_k}} \geq \frac{p_{g_k} - p_{j_1}}{w_{g_k} - w_{j_1}}.$$

That is, the relative value of g_k with respect to o_k is at least as high as the relative value of g_k with respect to j_1. Therefore, from (28),

$$\frac{p_{g_k} - p_{o_k}}{w_{g_k} - w_{o_k}} \geq \frac{p_s - p_t}{w_s - w_t} \quad \forall \quad k \in B. \tag{30}$$

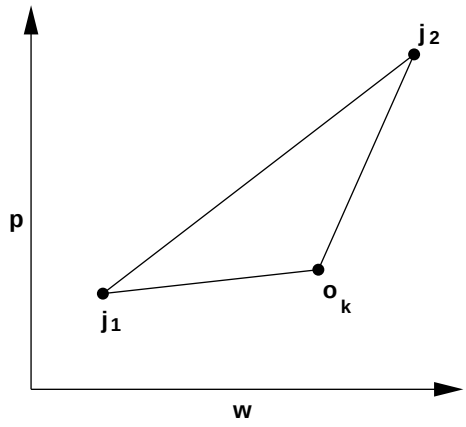


Figure 25: **Triangle formed by j_1, o_k and j_2.** Given that the relative value of j_2 with respect to j_1 is greater than or equal to the relative value of o_k with respect to j_1 (and knowing that $w_{j_1} \leq w_{o_k} \leq w_{j_2}$ and $p_{j_1} \leq p_{o_k} \leq p_{j_2}$) we deduce that the relative value of j_2 with respect to o_k is greater than or equal to the relative value of j_2 with respect to j_1.

4. Therefore, from (25), (26) and (30) we have

$$z \leq z^g + \sum_{k \in A} (w_{o_k} - w_{g_k}) \frac{p_s - p_t}{w_s - w_t} - \sum_{k \in B} (w_{g_k} - w_{o_k}) \frac{p_s - p_t}{w_s - w_t} \quad (31)$$

$$\leq z^g + \left[\sum_{k \in A} (w_{o_k} - w_{g_k}) - \sum_{k \in B} (w_{g_k} - w_{o_k}) \right] \frac{p_s - p_t}{w_s - w_t} \quad (32)$$

$$\leq z^g + \sum_{k=1}^{r} (w_{o_k} - w_{g_k}) \frac{p_s - p_t}{w_s - w_t}. \quad (33)$$

5. Let $\overline{c} = c - \sum_{i \in G} w_i$ be the space left in the knapsack after the selection of G, immediately before the rejection of the critical item s. From the fact that s was rejected we know that the difference in cost between s and t is greater than $\overline{c}$:

$$\overline{c} < w_s - w_t. \quad (34)$$

Furthermore we know that the total difference in cost between the items in the optimal solution and the items from the same candidate subsets in the greedy solution must be less than or equal to $\overline{c}$:

$$\sum_{k=1}^{r} (w_{o_k} - w_{g_k}) \leq \overline{c}.$$

Therefore, from (34),

$$\sum_{k=1}^{r}(w_{o_k} - w_{g_k}) \quad < \quad w_s - w_t. \tag{35}$$

6. Therefore, from (33) and (35),

$$\begin{aligned} z &\leq z^g + (w_s - w_t)\frac{p_s - p_t}{w_s - w_t} && (36)\\ &\leq z^g + p_s - p_t && (37)\\ &\leq z^g + p_s. && (38) \end{aligned}$$

Recall that the greedy algorithm compares the total profit of the final greedy solution (which is greater than or equal to z^g) to the total profit z^s of the cheapest solution containing the critical item (which is clearly greater than p_s), and keeps whichever solution is better. That is, the algorithm's solution has profit $z^h \geq \max(z^g, p_s)$. Therefore, from (38),

$$z^h \quad \geq \quad \frac{1}{2}z$$

and the profit of the algorithm's solution is guaranteed to be at least half the profit of the optimal solution.

3.7 Advantages and Limitations

Having proved that the solution of the full algorithm is always at least half-optimal, in this section we show that in the majority of practical cases its solution is much better than half-optimal. The same arguments apply to the simplified algorithm, for the subproblem of the MCKP in which the convexity assumption holds.

Recall that the maximum error of the algorithm is bounded by $p_s - p_t$ (see equation 37). This means that as the granularity of the candidate items (with respect to the total size of the knapsack) becomes finer, the maximum error tends to zero. The algorithm can be expected to perform much better than half-optimal for MCKP instances in which the granularity of the items is relatively fine, and pathological cases arise only when the difference in profit between s and t is a significant proportion of the optimal solution value z.[6]

We note that in level of detail applications especially it is rare that a single item or group of items will contribute a large proportion of the total profit of the optimal solution. In most level of

[6] A similar observation is made for other Knapsack Problem heuristics in [21].

detail applications hundreds or even thousands of scene objects are visible at the same time and each contributes a relatively small proportion to the total visual effect. The half-optimality guarantee is a worst-case figure and we expect the behaviour of the algorithm to be much better than half-optimal in most real-world applications.

Recall that the simplified algorithm depends on the convexity assumption for its half-optimality. However the IP-dominance of items with lower profit and greater cost than other items from the same candidate subset imply that the worst-case half-optimality of the algorithm is broken only for instances in which there exist items of *greater cost and profit* but *lower relative value* than other items from the same subset. These items are those that do not provide *diminishing returns*. Real world problem instances in which increased outlays do not provide diminishing returns are uncommon, as they tend to simplify the problem. All else being equal, we would always choose to select a more expensive item over less expensive ones that provide proportionately poorer returns. In level of detail situations it is rare that a more expensive representation of an object will provide proportionately greater perceptual benefit than a cheaper one. For example the successive addition of more polygons to a mesh representation of an object generally results in progressively smaller increases in visual quality, particularly if the additions are made in an intelligent "most significant first" fashion.

Note that in the case of the Continuous Multiple Choice Knapsack Problem C(MCKP) (see Definition 2.4 in Section 2.5.2) "concave" items — items that provide lower relative value than other more expensive items from the same candidate subset — are *LP-dominated* and can be removed from consideration completely [49]. A solution containing an LP-dominated item can be improved trivially by replacing that item with a linear combination of two items that lie on the convex boundary. By contrast the inability to select fractional portions of items in the MCKP means that these items must still be considered on the off chance that the items that dominate them cannot be afforded. The convexity assumption normally has no effect; non-convex items can be ignored completely *except* in the cases where the lowest cost items with cost greater than theirs for which the graph is convex are critical or post-critical. Therefore the simplified algorithm can be expected to perform well in most practical cases.

3.8 Comparison with Funkhouser-Séquin Algorithm

In this section we compare our greedy algorithms for the MCKP with that proposed by Funkhouser and Séquin (see Sections 2.5.4 and 2.6.1). In particular we discuss the practical implications of our

algorithm's proved correctness.

Recall from our discussion in Sections 2.5.4 and 2.6.1 that Funkhouser and Séquin proposed a greedy algorithm for the MCKP and pressed it into service (in an equivalent incremental form) as a non-hierarchical level of detail optimization algorithm. Recall also that in Definitions 2.5 and 2.6 we presented counterexamples for the claimed half-optimality of the Funkhouser-Séquin MCKP algorithm and incremental level of detail optimization algorithm.

Apart from the fact that our greedy algorithm always selects a feasible solution consisting of exactly one item from each subset, the main difference between our algorithm and theirs is that our algorithm is based on *relative value* rather than *value*. This is the feature that guarantees the half-optimality of our algorithm (for a simplified subproblem, in the case of the simplified algorithm). As we described in Section 3.1, the idea behind it is that the choice of which item to replace is based on the *relative* desirability of the potential replacement items (with respect to the items they would replace) rather than on their absolute value. This allows our algorithm to avoid inappropriately replacing items with other items that provide slight improvements in profit at the expense of much greater increases in cost when other replacement items, with lower absolute value but higher relative value, are available.

Examples of practical level of detail situations in which this may occur, as we noted in Section 2.6.1, are those in which objects have low detail impostor representations of barely significant cost that closely resemble their expensive high detail representations. For example, the low detail impostor might be a single texture mapped polygon and the high detail impostor a complex model consisting of hundreds of polygons. In these cases the higher detail impostors of objects provide slight increases in profit at the expense of dramatic increases in cost, and if they are selected the rendering time wasted is unavailable for improved renderings of other objects. Our algorithm is capable of taking this into account and favouring replacements of lower value but higher *relative* value instead, should they be available. The expensive representations are not ignored completely but are selected only when their selection is appropriate.

3.9 Incremental Version

Recall that our interest in efficient approximation algorithms for the MCKP arises from the fact that the MCKP is equivalent to the non-hierarchical level of detail optimization problem (as described in Chapter 2). The main limitation of our MCKP greedy algorithm, in this regard, is that it performs

a complete optimization from scratch for every frame, and so fails to exploit the considerable coherence that typically exists between successive frames. A more effective approach is to modify the algorithm to accept as input an initial best-guess solution derived from the previous frame.

Recall from Section 2.6.1 that Funkhouser and Séquin [24] propose an incremental version of their MCKP greedy algorithm that they claim is equivalent to it for the subproblem of the MCKP in which items with higher cost than other items from the same candidate subset always have lower *value*. This constitutes a *value-based convexity assumption* that is analogous to our relative-value-based one (introduced in Section 3.1). It is this assumption that allows their algorithm to be made incremental: it guarantees that the solution found by considering all items in descending order of value is the same as that found by considering, at each stage, only the items of immediately higher cost than those that are currently selected (from each candidate subset). This is because the items of immediately higher cost are guaranteed to have values that are at least as high as later (more expensive) items from the same subset and can therefore safely be assumed not to falsely poorly advertise them. If the convexity assumption does not hold then the incremental algorithm runs the risk that by considering and not selecting only the immediately higher cost items at each stage it will miss more expensive items that represent far better choices.

Recall that we showed in Section 2.5.4 that the Funkhouser-Séquin non-incremental MCKP greedy algorithm is not guaranteed half-optimal (even for the subproblem in which the value-based convexity assumption holds; note that the second counter example, presented in Definition 2.6, satisfies it). Likewise we showed in Section 2.6.1 that their equivalent incremental algorithm suffers from the same limitation.

In the same way that the value-based convexity assumption of Funkhouser and Séquin allows their greedy MCKP algorithm to be made incremental, so our relative-value-based convexity assumption allows our simplified algorithm to be made incremental. This is an important advantage of the simplified algorithm that provides another reason to prefer it for level of detail optimization. For this reason we choose to extend the simplified version in later chapters and have presented the full algorithm only for completeness.

We do not present the incremental version of our simplified MCKP algorithm here; rather we note that it is to our greedy algorithm what the Funkhouser and Séquin incremental algorithm is to theirs, and describe instead (in Chapter 7) an incremental version of a greedy algorithm for the *hierarchical* level of detail optimization problem that is a hierarchical extension (presented in Chapter 6) of our simplified MCKP greedy algorithm.

Note that while Funkhouser and Séquin exploit their value-based convexity assumption in the

making of their incremental algorithm, they do not formulate a simplified version of their MCKP algorithm designed to act more efficiently on a simplified subproblem. We exploit both benefits of the convexity assumption by formulating a simplified algorithm that is half-optimal as long as the assumption holds and then an incremental version that is equivalent to the simplified algorithm. The fact that the same convexity assumption provides both benefits is a fortunate coincidence, although of course the reason is the same in both cases.

3.10 Summary

In this chapter we presented a greedy approximation algorithm for the Multiple Choice Knapsack Problem (MCKP). This algorithm is based on a metric, *relative value*, that measures the profit density of candidate items with respect to other items from the same candidate subset. The relative value metric allows our algorithm to gauge effectively the "desirability" of items as replacements for previously selected items, thereby coping with the characteristic constraint of the MCKP that exactly one item must be selected from every candidate subset.

We proved that our algorithm's solution is always at least half as good as the optimal solution, and showed that the performance of the algorithm is much better than half-optimal in typical instances of the MCKP in which the granularity of the candidate items is fine with respect to the size of the knapsack. The time complexity of our algorithm is $O(n(m + r))$. The greedy algorithm, like that for the 0-1 KP described in Chapter 2, represents a compromise between optimality and efficiency. It provides a generally non-optimal solution of a guaranteed quality to an NP-complete problem in exchange for manageable polynomial time complexity.

In addition we presented a simplified version of the algorithm that exploits an assumption, called the *convexity assumption*, about the nature of the MCKP. We proved that this simplified algorithm's solution is always at least half-optimal for instances of the subproblem of the MCKP defined by the convexity assumption and showed that the convexity assumption is satisfied in the majority of useful problem instances. This simplified algorithm has a lower time complexity of $O(n \log n)$ and has the important additional advantage that it may be made *incremental*. This means that it may be modified to create an equivalent incremental algorithm that accepts as input an initial best-guess solution derived from the application of the algorithm to the previous problem instance. This allows the algorithm to exploit coherence between successive problem instances.

In later chapters we will put the advantages of the simplified algorithm to use in order to propose an incremental hierarchical level of detail optimization algorithm that is an extension of it. In the

meantime the next chapter, Chapter 4, presents a formal definition of a hierarchical level of detail description to serve as a basis for an investigation of the hierarchical level of detail optimization problem.

Chapter 4

Hierarchical Level of Detail Optimization

In Chapter 2 we described how the Multiple Choice Knapsack Problem (MCKP — see Section 2.5.2) had been shown to be equivalent to the level of detail optimization problem. Recall that in Section 2.6 we showed that this equivalence is broken by hierarchical level of detail descriptions with shared representations for groups of objects (such as those described by Maciel and Shirley [47], Chamberlain *et al* [14] and Shade *et al* [75]). We demonstrated that the few predictive hierarchical level of detail optimization schemes presented so far (namely those of Maciel and Shirley [47] and Belblidia *et al* [8]) have failed to address this problem completely. In this chapter we construct a basis on which to formulate our work by presenting a formal and very general definition of a *hierarchical level of detail description* and identifying the level of detail optimization problem that this description presents. We call this problem the *hierarchical level of detail optimization problem*.

If the characterizing feature of level of detail scene descriptions is that multiple representations, or *impostors*, may be provided for scene objects, the characterizing feature of hierarchical level of detail descriptions is that *shared* representations may be provided for groups of objects. The advantage of this is that rendering cost may be saved by the rendering of simple shared representations for groups of unimportant objects, while still providing a complete scene representation in which group object representations are coherent and consistent. Furthermore, hierarchies naturally allow the representation of the complex hierarchical structure of typical real-world scenes. Our definition attempts to capture these important characteristics.

Recall from Chapter 2 that the equivalence between the non-hierarchical level of detail optimization problem and the MCKP captures the idea that exactly one drawable representation must

be selected for each scene object. In contrast when an object in a hierarchical level of detail description is represented by a shared group representation of one of its ancestors, all of the objects which share that representation must necessarily be represented by it as well, as shown previously in Figure 12. There is no facility for taking this into account in the formalism of the Multiple Choice Knapsack Problem, where the selected items of distinct candidate subsets (corresponding to the selected impostors of distinct objects) are chosen independently. In this chapter we demonstrate that the level of detail optimization problem for a hierarchical level of detail description (which we shall refer to as the *hierarchical level of detail optimization problem*) is equivalent instead to a *hierarchical generalization* of the Multiple Choice Knapsack Problem in which the selection of items is limited by a hierarchy of constraints. We characterize this *Hierarchical Multiple Choice Knapsack Problem* formally in Section 4.2. In Chapter 5 we shall also present a new formalism for reasoning about such problems.

We begin in Section 4.1 by defining our level of detail hierarchy. In that section we will define hierarchical generalizations of useful concepts such as levels of detail and the incrementations and decrementations between them. In Section 4.2 we present a transformation of the hierarchical level of detail description defined in Section 4.1 to an equivalent *constrained* non-hierarchical one in which explicit constraints take the place of the implicit constraints represented in the structure of the hierarchy. We refer to this transformation in order to show the equivalence of the hierarchical level of detail optimization problem to the Hierarchical Multiple Choice Knapsack Problem. In Section 4.3 we revisit the hierarchical level of detail optimization algorithm of Maciel and Shirley (first discussed in Section 2.7.1) in light of this insight, evaluating its solution in terms of the Hierarchical MCKP. Finally we conclude this chapter in Section 4.4.

4.1 Hierarchical Level of Detail Description

An *object* is defined recursively as consisting of other objects that are its children or *parts*. The entire scene is represented by a hierarchy of such objects whose root is called the *scene object*. Each object may optionally be provided with a set of *impostors*, or drawable representations, that represent it and therefore all of its parts. The leaf objects must each be provided with at least one impostor. Where multiple impostors are associated with a single object, they are ordered uniquely according to increasing detail. The impostors of the parts of objects together form more detailed representations of those objects. Figure 26 shows an example of a simple level of detail hierarchy.

Note that this definition of a hierarchical level of detail description is fairly general. All of

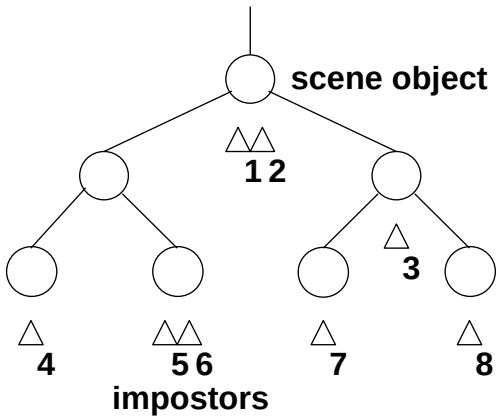


Figure 26: **Simple level of detail hierarchy.** Objects are represented by circles, and their impostors by triangles. The multiple impostors of each object are shown in order of increasing detail from left to right. The impostors of the descendants of each object constitute higher levels of detail of parts of that object. Impostors are numbered for convenience.

the hierarchical level of detail descriptions used by previous level of detail schemes (reviewed in Section 2.4) can be viewed as specialized instances of this description; in all of them shared simple representations (and sometimes multiple shared simple representations) are provided for groups of related objects. Different hierarchical descriptions differ from each other in the types of scene objects to which the "objects" in the hierarchy correspond, the ways in which they are combined to form group objects, the types of drawable geometric descriptions comprising the impostors, and the number of impostors allowed for each object.

4.1.1 Levels of Detail

We define a formal hierarchical generalization of the concept of a *level of detail*. Objects have multiple hierarchically defined levels of detail consisting of both their own *explicit* impostor representations and the *implicit* representations consisting of the combinations of the impostors of their descendants. Each level of detail corresponds to a unique set of selected impostors:

Definition 4.1 *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* s.

For example a valid level of detail of the scene object in Figure 27 is the set of impostors $\{4, 6, 3\}$. This definition ensures that each level of detail of an object provides some complete and unambiguous representation of every part of that object; be it one of its associated impostors or a subset of the impostors of its descendants. In addition objects that are parts of other objects may also be represented instead by the impostors of their ancestors.

4.1.2 Replacement Sets

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

Definition 4.2 *Replacement Set*

The replacement set *of an impostor i 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 descendants of O, otherwise.*

Figure 27 illustrates examples of various replacement sets in a simple 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 "root" impostor corresponding to *no representation*.

4.1.3 Incrementation and Decrementation

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 i 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.

4.1.4 Partial Ordering of Levels of Detail

The levels of detail of each object are partially ordered by the following relation:

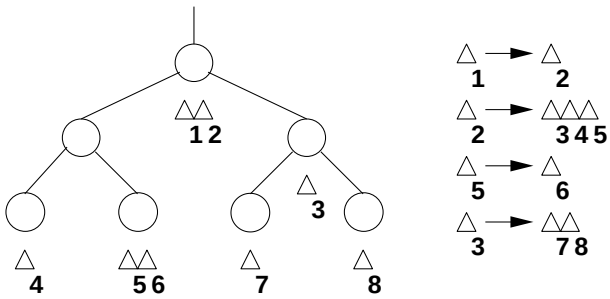


Figure 27: **Examples of replacement sets.** The replacement set of impostor 1 is $\{2\}$, and that of impostor 2 is $\{4, 5, 3\}$. The replacement set of impostor 5 is $\{6\}$ and that of 3 is $\{7, 8\}$. Impostors 4, 6, 7 and 8 have no replacement sets, and $\{1\}$ is the replacement set of an imaginary impostor *no representation*.

Definition 4.3 *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 $i \in \{1, 2, 3, \ldots, n-1\}$.

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 levels of detail that are strictly lower and strictly higher, respectively. Figure 28 illustrates the partial ordering of levels of detail.

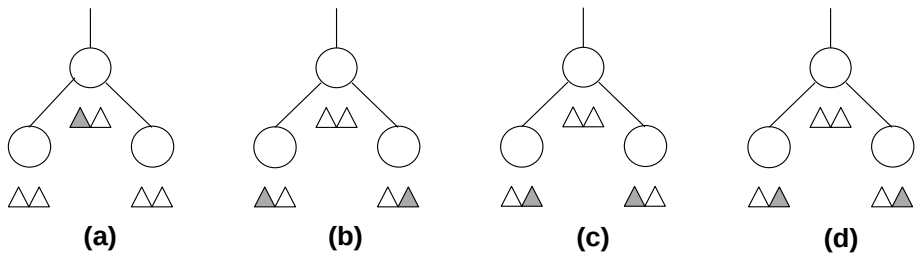


Figure 28: **Partial ordering on levels of detail.** Four levels of detail of a simple level of detail hierarchy. Level of detail a is the lowest level of detail of the hierarchy, and d is the highest. Levels of detail b and c are related to a and d by $a \leq b \leq d$ and $a \leq c \leq d$. However b and c are not related: although they are not equal, neither is a higher or lower level of detail than the other.

Even if a level of detail s of an object O is strictly lower than another level of detail 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 4.4 *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 $s \cap t = \emptyset$.*

4.1.5 Covering of Replacement Sets

In the case of *non-hierarchical* level of detail descriptions there is a sense in which a particular level of detail of the whole description may contain a higher or lower detail impostor of a particular object than one of that objects other impostors. We say that an impostor i of an object O is *covered* by some level of detail s of the entire description if the impostor of O contained in s is greater than or equal to i. This terminology is useful in comparing levels of detail with impostors of particular objects. We can define a similar concept in the case of hierarchical level of detail descriptions, but we must talk of replacement sets rather than of impostors, and of the levels of detail of objects rather than of the entire description:

Definition 4.5 *Covering of Replacement Sets*

Let R be the replacement set of some impostor of an object O. We say that R is covered *by a level of detail s of some 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, during which R is replaced by its higher detail representation embodied in t.

4.1.6 Ancestor and Descendant Replacement Sets

Lastly, we define two terms that reflect a partial ordering of replacement sets:

Definition 4.6 *Ancestor Replacement Sets*

We say that a replacement set R is an ancestor replacement set *of another replacement set S if there exists a (possibly empty) list of replacement sets $R_1, R_2, R_3, \ldots, R_n$ such that $R = R_1$, $S = R_n$, and R_{i+1} is the replacement set of some impostor in R_i for $i \in \{1, 2, 3, \ldots, n-1\}$.*

Definition 4.7 *Descendant Replacement Sets*

S *is a* descendant replacement set *of* R *if* R *is an* ancestor replacement set *of* S.

In the example shown in Figure 27, $\{3, 4, 5\}$ is a descendant replacement set of $\{2\}$ and an ancestor replacement set of $\{6\}$ and $\{7, 8\}$. Note that all replacement sets are trivially ancestors and descendants of themselves.

4.2 Hierarchical Multiple Choice Knapsack Problem

In the hierarchical level of detail scene description defined in Section 4.1, each group (or non-leaf) object is the union of its parts, or children. Therefore impostors of group objects are essentially shared representations of all of the parts of those group objects. By our definition they function as lower detail representations of those parts than any of the impostors that are explicitly associated with the parts themselves. We may therefore redraw the hierarchy equivalently by transforming group object impostors into shared low-detail impostors of their children, as long as we note that the shared impostors are constrained and must be selected in unison for all of the parts, if at all (see Figure 29).

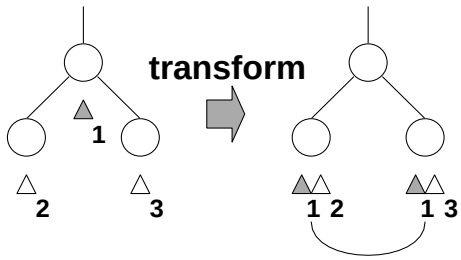


Figure 29: **Transformation of a group object impostor.** Impostors of group objects may be equivalently regarded as shared impostors of the children of those group objects. For clarity, each object is assumed to have exactly one impostor, impostors are numbered, and the inherited impostor of the group object is shaded. The link attached to the shared impostor indicates that the objects which share it must take it on in unison.

By repeatedly applying this transformation we may create an "empty" or flat hierarchy with impostors only at the leaves (see Figure 30). Each leaf has as its impostors all of its own impostors plus a series of lower detail impostors inherited in top-down order from its ancestors in the hierarchy. The equivalence is subject to a set of *constraints*, one for each original group impostor: the leaf objects

that share each inherited group impostor *must take on that shared impostor in unison*. The resulting flat hierarchy is essentially a hierarchically constrained *non-hierarchical level of detail description* (Section 2.6), exactly equivalent to the original hierarchical one. The immediately higher impostors of the objects that share each inherited group impostor together constitute the replacement set of that impostor.

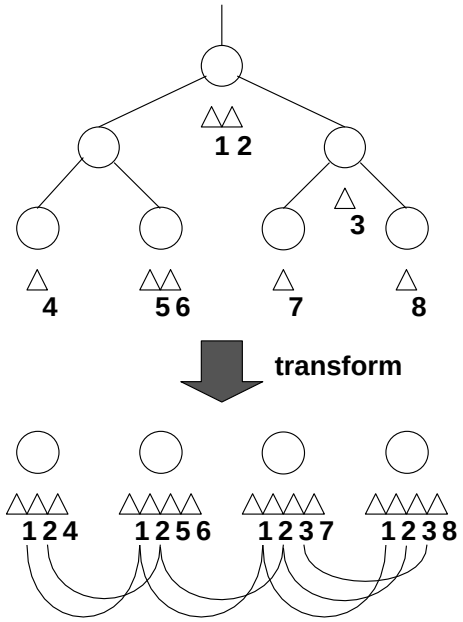


Figure 30: **Transformation of a simple level of detail hierarchy.** The impostors of group objects have all been transformed into shared impostors of the leaf objects. The constraints between shared impostors, shown as links, imply that the objects which share those impostors must take them on in unison.

Using the transformation described above, we can now show that the hierarchical level of detail optimization problem is equivalent to a *hierarchical generalization* of the Multiple Choice Knapsack Problem, shown conceptually in Figure 31. We refer to this constrained MCKP as the *Hierarchical Multiple Choice Knapsack Problem*, in reference to the fact that the constraints are hierarchical in nature and make explicit the implicit constraints represented by the structure of a hierarchical level of detail description. The candidate items correspond to the impostors of the objects in the equivalent non-hierarchical description, and are divided into candidate subsets such that each subset corresponds to the impostors of a single object. At most one item may be selected from each candidate subset.

The Hierarchical MCKP differs from the usual MCKP in that some items are members of more than one candidate subset: those which correspond to shared group impostors in the non-hierarchical description. These correspond to impostors of more than one object, and their corresponding items are capable of representing several candidate subsets at a time. The hierarchical structure of the constraints corresponds, in our case, to the hierarchical structure of the level of detail description. Note that the standard MCKP is a special case of the Hierarchical MCKP, just as a non-hierarchical level of detail description is a special case of the more general hierarchical level of detail description. The Hierarchical MCKP is therefore NP-complete.

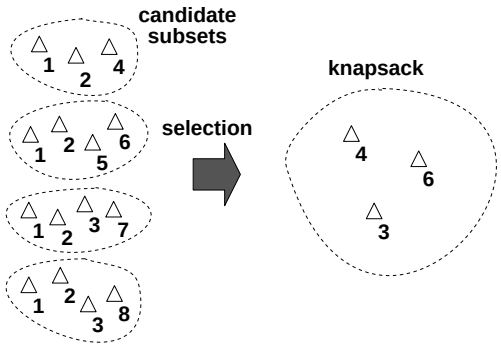


Figure 31: **Hierarchical Multiple Choice Knapsack Problem.** The Hierarchical MCKP is identical to the MCKP except that some candidate items are elements of more than one candidate subset. These shared items are *constrained* and may only be selected together. Compare with Figure 7.

We define the *replacement set* of an item in an instance of the Hierarchical MCKP to correspond exactly to the replacement set of the impostor in the hierarchical level of detail description to which the instance corresponds. The items in the replacement set correspond to the impostors in the replacement set of the corresponding impostor.

The definition of the Hierarchical MCKP is identical to that of the MCKP given in Definition 2.3 (See Section 2.5.2) except that the candidate subsets are not necessarily disjoint. We allow each item to have a single replacement set and require that for every candidate subset of which an item is an element, exactly one item in its replacement set must be an element of that subset. In addition we require that all replacement sets are disjoint.

Definition 4.8 *The Hierarchical Multiple Choice Knapsack Problem*

Given a set N *of* n candidate items, *a set of* r candidate subsets $N_1, \ldots, N_r$, *and a* knapsack,

with

$$\begin{aligned} p_j &= \text{profit } \textit{of item } j \\ w_j &= \text{cost } \textit{of item } j \\ c &= \text{capacity } \textit{of the knapsack} \end{aligned}$$

maximize

$$z = \sum_{j=1}^{n} p_j x_j$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} w_j x_j &\leq c \\ \sum_{j \in N_k} x_j = 1 \quad &\forall \quad k \in \{1, \ldots, r\} \\ x_j \in \{0,1\} \quad &\forall \quad j \in N \\ N &= \{1, \ldots, n\} = \bigcup_{k=1}^{r} N_k \end{aligned}$$

and where the root item $o \in N_k, k = 1, \ldots, r$ *has a* replacement set R_o *where the replacement set of an item* i *is a set of items* $R_i = \{i_1, i_2, i_3, \ldots, i_{z_i}\}$ *such that:*

1. *For each candidate subset* N_k *of which* i *is an element, there exists exactly one item* $j \in R_i$ *that is an element of* N_k*.*
2. *Each item* $j \in R_i$ *may or may not have a replacement set.*
3. *All replacement sets are mutually disjoint.*

4.3 Maciel-Shirley Algorithm Revisited

Having produced a formal description of the hierarchical level of detail optimization problem in the form of the Hierarchical MCKP defined in Section 4.2, we are now in a position to reconsider more critically the hierarchical level of detail optimization algorithm of Maciel and Shirley [47], which we discussed earlier in Section 2.7.1. Being a predictive level of detail optimization algorithm for a hierarchical level of detail description similar to that defined formally in Section 4.1, it is an

approximation algorithm for the hierarchical level of detail optimization problem and therefore for the Hierarchical MCKP.

The main characterizing feature of the Maciel and Shirley algorithm is that is also a greedy algorithm. Whereas we in our greedy algorithm for MCKP use *relative value* (Section 3.1) and Funkhouser and Séquin use *value* (Section 2.5.4), Maciel and Shirley use *profit*, which they refer to as *importance*. Essentially the algorithm considers replacement sets as replacements for the items that they replace, favouring those replacement sets that represent more *important* objects. The profit of an impostor, for the purposes of replacement selection, is the importance of the object owning that impostor. The importance of an object in turn is defined for group objects as the importance of their most important part. The importance of leaf objects is defined as a weighted average of several factors *intrinsic* to objects such as their screen-space size, distance from the line of sight, relative speed and inherent semantic importance. This favouring of more detailed representations of more important objects constitutes a "best-first" [47] greedy selection of replacement sets. Care is taken to ensure the consistency and completeness of the solution (and therefore the scene representation) by selecting initially a single impostor representing the entire scene and replacing impostors with replacement sets that collectively represent exactly the same objects.

However, as we showed in Section 2.7.1, the algorithm of Maciel and Shirley is flawed and places no guarantee on the quality of its solution. This failing is a direct result of the use of profit (or importance) as the greedy heuristic rather than value (or more precisely, as we showed for the MCKP in Chapter 3 and shall show for the Hierarchical MCKP in Chapter 6, *relative value*). For any optimization problem any number of greedy heuristics are available; for example selection by maximum profit, minimum cost, maximum value and, as we have shown, maximum relative value. In the case of the 0-1 KP, only selection by maximum value is effective. It succeeds because it takes into account both profit and cost, selecting items that provide the best combination of the two. In the case of MCKP, as we saw in Chapter 3, selection by maximum relative value is the most successful heuristic, since it takes into account the profits and costs of not only the replacing item but also the replaced item. In the case of the Hierarchical MCKP, just as in the 0-1 KP and MCKP, selection by maximum profit is not the most effective heuristic. In Chapter 6 we show that the heuristic of choice is a hierarchical extension of *relative value*.

4.4 Summary

In this chapter we have presented a formal definition of a generalized hierarchical level of detail description. This description is characterized by the fact that multiple shared representations may be provided for groups of related scene objects. Our formalized definition captures the levels of detail that such a hierarchical description can reflect, and the nature of their ordering. Furthermore we have derived operations, *incrementation* and *decrementation*, that transform one level of detail to another, and terms that allow us to speak about the relationships between levels of detail and object representations.

Having defined formally a hierarchical level of detail description, we showed how such a description can be transformed to an equivalent *constrained non-hierarchical* one in which group object impostors are explicitly shared between the parts of those group objects. The constraints on shared impostors implied by the original hierarchical structure are represented explicitly in the form of constraints. We demonstrated, by referring to this equivalent constrained non-hierarchical description, that the hierarchical level of detail optimization problem is equivalent to a hierarchical generalization of the Multiple Choice Knapsack Problem, which we call the *Hierarchical Multiple Choice Knapsack Problem*. This Hierarchical MCKP differs from the conventional MCKP in that candidate items may belong to more than one candidate subset. In the following chapters we will derive formalisms for dealing with the Hierarchical MCKP.

Chapter 5

Level of Detail Graphs

In Chapter 4 we presented a formal description of a generalized hierarchical level of detail description whose characterizing feature is that multiple shared drawable representations, or *impostors*, may be provided for groups of scene objects. It is clear that such hierarchical level of detail descriptions give rise to large numbers of *levels of detail* that are the valid combinations of selecting impostors from within the hierarchy and differ from one another by the selection or deselection of particular impostors. These levels of detail are connected to one another by *incrementations* and *decrementations* that function as transformations from one level of detail to another. Each hierarchical description gives rise to its own *state space* of levels of detail, which can be visualized as a connected graph. In particular it is a *lattice*, due to the partial ordering on levels of detail derived in Section 4.1.4.

In this chapter we investigate these level of detail state spaces and provide a formalism in which they can be analyzed. We introduce the concept of a *level of detail graph* as a means of representing (visually, semantically and automatically in computer programs) the state spaces generated by hierarchical level of detail descriptions.[1] We will make use of these level of detail graph descriptions in Chapter 7 where we use them to prove the equivalence of two hierarchical level of detail optimization algorithms.

In Section 5.1 we provide an overview of level of detail graphs in their basic form. We expand on this by discussing the level of detail graphs of non-hierarchical level of detail descriptions in Section 5.2 and then hierarchical descriptions in Section 5.3. Finally we conclude the chapter in

[1] Note however that these level of detail graph representations may be applied equally well to the special case of purely non-hierarchical level of detail descriptions, where they form a special regular case of the more general irregular level of detail graphs.

Section 5.4 with a summary of the major points.

5.1 Level of Detail Graphs

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 4.3) 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.

5.2 Non-Hierarchical Level of Detail Descriptions

We first consider the level of detail graphs generated by non-hierarchical level of detail descriptions. Recall from Section 2.6 that a non-hierarchical level of detail description is a scene description in which multiple drawable representations may be provided for each scene object, but no shared representations may be provided for groups of objects. In these descriptions the replacement set of an impostor is always simply the immediately higher impostor of the same object, if one exists (See Definition 4.2 in Section 4.1.2). The level of detail graphs generated by non-hierarchical descriptions are all regular lattices — or grids — 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 32 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 two paths between the same two nodes involve the same series of replacements, although the ordering of the series is unique to each path.

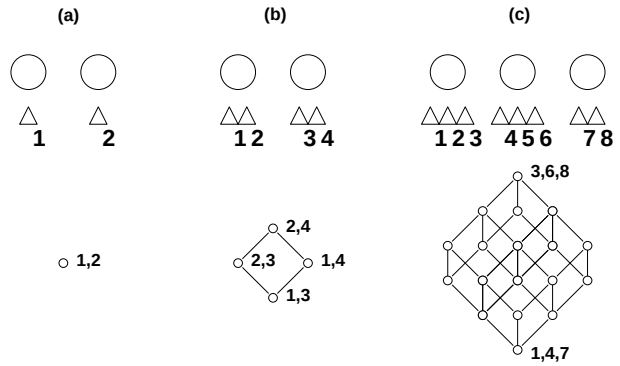


Figure 32: **Level of detail graphs of non-hierarchical descriptions.** Three simple non-hierarchical level of detail descriptions, numbered (a) to (c), and their corresponding level of detail graphs. Some nodes are unlabed in (c) for clarity.

5.3 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 4.2 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.

5.3.1 Single Constraint

The effect of introducing a single constraint is a well-defined change in the structure of the level of detail graph, as shown in Figure 33. 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. Recall that the replacement set of a shared impostor consists of the immediately higher impostors of the linked objects in the constrained non-hierarchical description (Section 4.2). The algorithm that applies the effects of a constraint on a level of detail graph is given in Figure 34. It is assumed in the description of the algorithm that the constraint being applied corresponds to a shared group impostor in a valid hierarchical level of detail description of the type defined in Chapter 4.

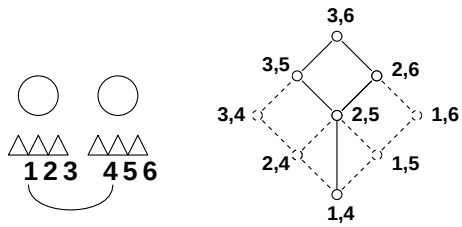


Figure 33: **Effects of a single constraint.** The effects of the application of a single constraint (on impostors 1 and 4) to a non-hierarchical level of detail description on the level of detail graph of the description. Removed states and arcs are shown with dotted lines. Note the addition of a new arc between state 1,4 and state 2,5.

Figure 35 compares the effects of two single constraints applied to a simple non-hierarchical level of detail description. Figure 35 (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).

5.3.2 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 30. There are nonetheless certain requirements that are satisfied by any constrained non-hierarchical description that is equivalent to a valid hierarchical description:

1. Each impostor may be constrained by at most one constraint.

2. Each constraint may constrain at most one impostor of each object.

3. 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 impostor then all lower impostors of that object are also inherited group impostors).

4. 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.

```
input: a level of detail graph G
input: a set of impostors C that are subject to the new constraint
input: R, the replacement set of the group impostor corresponding to C
output: the new level of detail graph G

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 ∩ s ≠ ∅ and C − s ≠ ∅ then
            remove s and all arcs incident to s 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 with R

        if C ⊆ s then
        {
            set t ← (s − C) ∪ R
            create a new arc from level of detail s to level of detail t in G
        }
    }
end
```

Figure 34: **The constraint algorithm.** The constraint algorithm takes as input a level of detail graph G, a set of impostors C that are subject to the new constraint, and the replacement set R of the group impostor corresponding to C. Its output is the level of detail graph after application of the constraint.

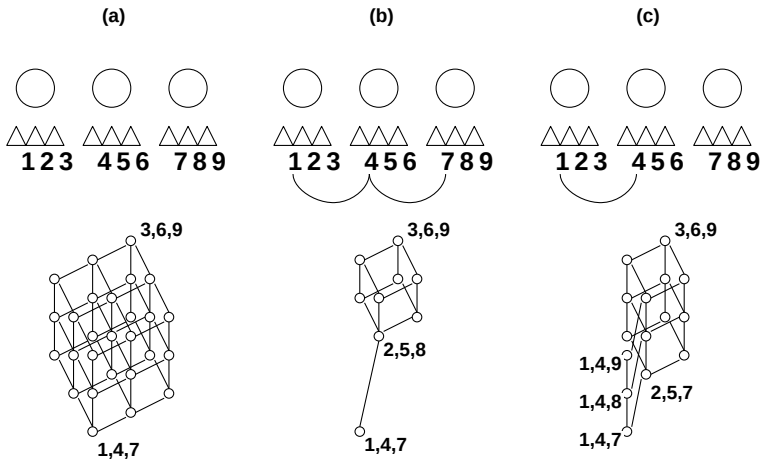


Figure 35: **Comparison of the effects of two single constraints.** The effects of two single constraints applied to a simple non-hierarchical level of detail description. The unconstrained non-hierarchical description and its regular 3-dimensional level of detail graph are shown in (a). In (b) the lowest impostors of the first two objects are constrained, and in (c) the lowest impostors of all three objects.

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 34 for each constraint in turn, in increasing order of detail of the impostors they constrain. The algorithm is shown in Figure 36.

Figure 37 shows an example hierarchical level of detail description, its equivalent constrained non-hierarchical description, and the generation of its corresponding level of detail graph.

5.4 Summary

Since we have introduced the novel notion of the Hierarchical Multiple Choice Knapsack Problem in Chapter 4, we have had to derive new methods of reasoning about the implications of hierarchical level of detail descriptions for level of detail optimization. In this chapter we have presented a new representation, called *level of detail graphs*, of the state spaces of hierarchical level of detail scene descriptions. This representation is general and may be applied to any hierarchical level of detail description or scene description that is described by the formal definition given in Chapter 4. We have provided algorithms for the generation of the level of detail graphs of arbitrary hierarchical

```
input: a non-hierarchical level of detail description
input: a set of constraints to be applied
output: the level of detail graph G of the constrained description

begin
        // begin with the level of detail graph of the unconstrained description
        // and apply the Constraint Algorithm for each constraint in order

    set G → level of detail graph of the unconstrained description
    for each object O of the non-hierarchical description
        for each impostor i of O from lowest to highest
            if there exists a constraint C containing i then
                if constraint C has not already been applied then
                    set G → result of applying C to G using the Constraint Algorithm
end
```

Figure 36: **Level of detail graph generation algorithm.** Algorithm to generate the level of detail graph of an arbitrary constrained non-hierarchical level of detail description (and therefore of a hierarchical level of detail description).

level of detail descriptions. These level of detail graph representations and their generation algorithms serve equally well for the representation of the state spaces of non-hierarchical descriptions, where they are a simpler special case.

Applications for these level of detail graph representations include the investigation of the behaviour and functionality of hierarchical and non-hierarchical level of detail optimization algorithms. They function as a graphical and semantic notation to visualize and simulate more easily the operation of such algorithms, which may be viewed as traversals of the level of detail graphs generated by their level of detail descriptions.

In the next chapter we return our attentions to the hierarchical level of detail optimization problem, devising a greedy approximation algorithm for the Hierarchical Multiple Choice Knapsack Problem based on the simplified greedy algorithm for the MCKP that we presented in Chapter 3.

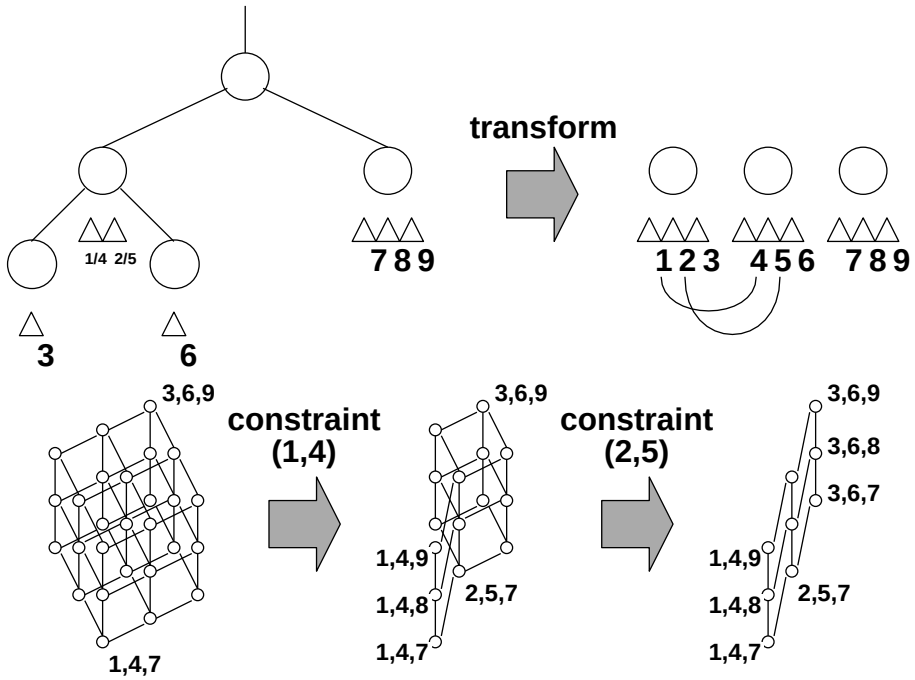


Figure 37: **Level of detail graph of a hierarchical level of detail description.** A hierarchical level of detail description, its equivalent constrained non-hierarchical description, and the generation of its level of detail graph. We begin with the graph of the unconstrained non-hierarchical description and apply the constraints $\{1, 4\}$ and $\{2, 5\}$ in that order. Impostors are numbered uniquely in the constrained non-hierarchical description for clarity. Group objects in the hierarchical description are numbered with the numbers of the shared impostors to which they correspond.

Chapter 6

Greedy Algorithm for the Hierarchical MCKP

In this chapter we present a greedy approximation algorithm for the Hierarchical Multiple Choice Knapsack Problem that we introduced in Chapter 4. In the description of the algorithm we make use of the concept of *replacement sets* that we defined in Section 4.1.2 for impostors and in Section 4.2 for items in the Hierarchical MCKP.

This greedy algorithm for the Hierarchical MCKP is a hierarchical extension of the simplified greedy algorithm for the conventional MCKP presented in Chapter 3. We prove that its solution to the Hierarchical MCKP is guaranteed to be at least half-optimal (in terms of total profit) for instances of a subproblem of the Hierarchical MCKP in which more detailed representations provide diminishing returns. The time complexity of the algorithm is $O(n \log n)$ with respect to the number of candidate items.

We begin in Sections 6.1 and 6.2 by defining hierarchical versions of the *relative value* metric and the convexity assumption defined previously in Chapter 3. In Section 6.3 we present the algorithm, and in Section 6.4 we prove its correctness. We discuss the limitations and advantages of the algorithm in Section 6.5, showing that, like the greedy algorithm for the MCKP presented in Chapter 3, its expected error is typically very small. In Section 6.6 we introduce the idea of making the algorithm incremental. Finally we close the chapter in Section 6.7 with a summary of the major points.

6.1 Hierarchical Relative Value

In Chapter 3 we defined a metric called *relative value* that measured the desirability, or *relative profit density*, of items relative to other items of the same candidate subset that they might replace (see Section 3.1). This relative value metric was put to use in the greedy algorithms for the MCKP described in Chapter 3. Recall that in general the relative value of an item may be defined relative to any other item from the same candidate subset, but because our simplified MCKP algorithm always considers the candidate items in each subset in ascending order of cost, in the case of that algorithm relative value was always defined relative to the immediately lower cost item from the same subset. Likewise in this algorithm for the Hierarchical MCKP we invariably measure the relative value of more expensive representations with respect to immediately less expensive ones. Recall from Chapter 4 that we defined the *replacement set* of an item in the Hierarchical MCKP to be the set of items that, in a sense, together constituted its immediately higher detail representation. Here we define a hierarchical version of *relative value* which measures the desirability of replacement sets relative to the items that they replace:

Definition 6.1 *Relative Value of a Replacement Set*

The relative value *of the replacement set R of an item i is measured with respect to i and is defined as follows:*

$$RV(R) = \frac{(\sum_{j \in R} p_j) - p_i}{(\sum_{j \in R} w_j) - w_i}$$

The relative value of replacement set R provides a measure of the advantage gained by using it to replace the item i that it replaces, and is used to compare replacement sets against replacement sets of other items. This is measured as the ratio of the differences in total profit and cost between the replacement set R and the replaced item i. The relative value measure enables us to gauge the desirability of potential replacement sets, just as the non-hierarchical relative value metric allowed us to measure the desirability of replacement *items* in Chapter 3.

6.2 Hierarchical Convexity Assumption

Our algorithm assumes for its half-optimality that the replacement set of an item will always have greater total cost than that item and lower relative value than the replacement set (if any) of which that item is an element (See Figure 38). If this requirement is satisfied by an instance of the Hierarchical MCKP, then the algorithm's solution to that instance is guaranteed to be half-optimal, as

we shall demonstrate in Section 6.4. This requirement is a hierarchical version of the *convexity assumption* introduced in Section 3.2.

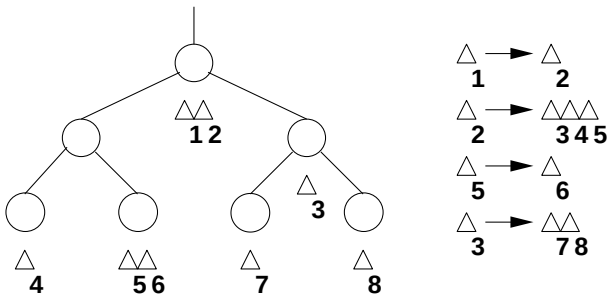


Figure 38: **The hierarchical convexity assumption.** Our algorithm requires for its half-optimality that replacement sets should always have lower relative value than the replacement sets (if any) containing the candidate items that they replace. In this example, this implies that the replacement set $\{3, 4, 5\}$ has lower relative value than the replacement set $\{2\}$ (that is, $\frac{(p_3+p_4+p_5)-p_2}{(w_3+w_4+w_5)-w_2} \leq \frac{p_2-p_1}{w_2-w_1}$), and replacement sets $\{6\}$ and $\{7, 8\}$ both have lower relative value than $\{3, 4, 5\}$.

The convexity assumption is likely to be satisfied in many instances of the Hierarchical MCKP that arise in level of detail optimization. It implies that higher detail representations of objects must provide *increased perceptual benefit* at the expense of *increased rendering cost*, with diminishing returns for increasingly more detailed representations. We shall discuss this further in Section 6.5.

6.3 Greedy Algorithm

In this section we present our greedy algorithm for the Hierarchical MCKP, shown in Figure 39. The algorithm accepts as input an instance of the Hierarchical MCKP and produces as output a feasible solution to that instance. The algorithm begins with the simplest feasible solution and iteratively replaces items with their replacements sets as far as the available cost will allow. It maintains the feasibility of the solution by always replacing an item with its complete replacement set and ensuring that replacement sets are only considered for selection when the items they replace have already been selected. It maximizes the quality of the solution by favouring, when given the choice, replacements that result in the greatest increase in profit for the smallest increase in cost. In order to determine the most desirable replacements, the algorithm makes use of a simple selection heuristic based on the hierarchical relative value metric defined in Section 6.1.

input: an instance of the Hierarchical MCKP (see definition 4.8)
output: a feasible solution to that instance

begin
  set $S \leftarrow \emptyset$, $L \leftarrow$ empty list // no critical replacement set yet, empty list
  for each item i in the replacement set of *no representation*
  {
    set $x_i \leftarrow 1$ // select the cheapest feasible solution
    set $R \leftarrow$ the replacement set, if any, of i
    insert R into L in descending relative value order
  }
  while L is not empty // while there are unconsidered replacement sets
  {
    set $R \leftarrow$ the first replacement set in L
    set $i \leftarrow$ the item whose replacement set is R
    remove R from L
    if $\sum_{i \in N} x_i w_i - w_i + \sum_{i \in R} w_i \leq c$ **then** // if we can afford to replace i with R
    {
      set $x_i \leftarrow 0$ // unselect i
      for each item $j \in R$
      {
        set $x_j \leftarrow 1$ // select every item in R
        if j has a replacement set, T, **then**
          insert T into L in descending relative value order
      }
    }
    else if $S = \emptyset$ **then set** $S \leftarrow R$ // if no critical replacement set yet, R is critical
  }
  if $S \neq \emptyset$ **then** // if there is a critical replacement set
  { // then consider the critical solution
    find H, the lowest cost critical replacement set solution
    if $\sum_{i \in H} p_i > \sum_{i \in N} x_i p_i$ **and** $\sum_{i \in H} w_i \leq c$ **then**
      set $x_i \leftarrow \begin{cases} 1 & \text{if } i \in H \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in N$
  }
end

Figure 39: **Greedy algorithm for the Hierarchical MCKP.**

The algorithm maintains a list of replacement sets currently available for selection, ordered by descending relative value. It initially selects the imaginary *root item* (See Definition 4.8 in Section 4.2) that has no profit and cost and is an element of every candidate subset, and inserts into the replacement set list the replacement set of this item. It then greedily considers the remaining replacement sets in order of descending relative value, using the replacement set list to ensure that each replacement set is only considered after the item it replaces has been selected. While the replacement set list is not empty, the algorithm considers the first replacement set in the list for replacement, substituting it for the item it replaces if this replacement can be afforded. If the replacement is made then the replacement set is removed from the replacement set list and the replacement sets (if any) of the items it contains are inserted into the list in descending relative value order. Otherwise the replacement set is simply removed from the list and discarded. If it is the first replacement set to be so discarded it is marked as the *critical replacement set*. At any stage the replacement sets that are available in the replacement set list are those whose associated items are currently selected, and they are considered in descending order of relative value.

When this greedy selection terminates (due to the replacement set list being found to be empty) the solution reached is compared against the lowest cost feasible solution containing the critical replacement set. If this *critical replacement set solution* has greater profit than the greedy selection and has total cost less than or equal to the size of the knapsack then it is selected instead.

Finding the critical replacement set is simple: it is found as a by-product of the greedy selection stage. Finding the lowest cost feasible solution containing the critical replacement set however requires an iterative selection process similar to the greedy selection stage but taking no notice of relative value orderings and selecting only replacement sets that represent candidate subsets that are also represented by the critical replacement set. An algorithm for finding the lowest cost critical replacement set solution is shown in Figure 40.

Our Hierarchical MCKP algorithm essentially favours the more detailed representations of selected items whose more detailed representations have greatest relative value. These are those that provide the greatest increase in profit for the smallest increase in rendering cost when replacing their immediately lower cost representations. By considering more detailed representations in units corresponding to replacement sets, the algorithm ensures that when a shared item is replaced it is replaced completely by its entire replacement set. In this way the algorithm preserves the feasibility of the approximate solution, since every replacement set represents the same candidate subsets as the item that it replaces. The algorithm therefore always produces a feasible solution in which exactly one item is selected from each candidate subset.

```
input: an instance of the Hierarchical MCKP (see Definition 4.8)
input: the critical replacement set for that instance, S
output: the lowest cost feasible solution containing the critical replacement set

begin
    set L ← empty list of replacement sets
    for each item i in the replacement set of no representation
    {
        set x_i ← 1                          // select the cheapest feasible solution
        set R ← the replacement set, if any, of i
        append R onto the end of L
    }
    while L is not empty                     // while there are unconsidered replacement sets
    {
        set R ← the first replacement set in L
        set i ← the item whose replacement set is R
        remove R from L

            // if R represents all the replacement sets that S does
        if there exists an item j ∈ R, j ∈ N_k for every i ∈ S, i ∈ N_k for some k then
        {
            set x_i ← 0                      // unselect i
            for each item j ∈ R
            {
                set x_j ← 1                  // select every item in R
                if j has a replacement set, T, then
                    append T onto the end of L
            }
        }
    }
end
```

Figure 40: **Critical replacement set solution algorithm.** This algorithm may be used to find the cheapest feasible solution containing the *critical replacement set*, which is needed by the Hierarchical MCKP algorithm of Figure 39.

6.4 Proof of Half-Optimality

In this section we prove the half-optimality of the greedy algorithm described in Section 6.3, for the subproblem of the Hierarchical MCKP in which the hierarchical convexity assumption holds (see Section 6.1). Instances of this subproblem are those in which every replacement set has greater profit and cost than the item it replaces and lower relative value than the replacement set (if any) containing that item. We assume in this proof that those requirements are satisfied.

The proof is a hierarchical extension of that already provided for the simplified greedy algorithm for the MCKP in Section 3.4. This proof differs from that for the simplified MCKP algorithm due to the fact that in the Hierarchical MCKP we deal with replacement sets rather than replacement items. Likewise relative value is defined for replacement sets (as described in Section 6.1) rather than for relative items, and the proof takes this into account.

6.4.1 Overview of Proof

Like the proofs for the simplified and full MCKP algorithms presented in Sections 3.4 and 3.6, the proof is composed of six steps:

1. We formulate an equation relating the profit of the optimal solution to that of an intermediate stage in the greedy solution corresponding to the solution reached up to the consideration and rejection of the critical item. This equation provides an upper bound on the maximum error of the greedy algorithm and includes terms that quantify the profit gained by selecting items that are not in the optimal solution and the profit lost by not selecting items that are.

2. We show that any replacement sets that are in the optimal solution but were not selected by the greedy algorithm before the rejection of the critical replacement set must have lower relative value than the critical replacement set, since they were not considered before it.

3. Similarly we show that any replacement sets that were selected by the greedy algorithm before the rejection of the critical replacement set but are not in the optimal solution must at least have higher relative value than the critical replacement set, since they were considered before it.

4. Using the results of steps 2 and 3, we show that the maximum error of the greedy algorithm is bounded by the total difference in cost between the optimal solution and the intermediate greedy solution, multiplied by the relative value of the critical replacement set.

5. Then we show that the difference in cost between the optimal solution and the intermediate greedy solution is bounded by the difference in cost between the critical replacement set and the item that it would replace.

6. Substituting, we show that the maximum error of the intermediate greedy solution is bounded by the difference in profit between the critical replacement set and the item that it replaces. Recalling that the algorithm compares the total profit of the final greedy solution (which is necessarily greater than the profit of the intermediate solution) to the profit of the cheapest feasible solution containing the critical replacement set and keeps whichever is better, we conclude that the algorithm's solution is at least half as good as the optimal one.

6.4.2 Proof

1. Given an instance of the Hierarchical MCKP, let the profit of the optimal solution to this instance be z. Let G be the set of items in the intermediate solution reached by the greedy algorithm immediately before the critical replacement set is considered (and rejected), and let $z^g = \sum_{i \in G} p_i$ be the profit of this intermediate greedy solution.[1]

   Therefore

   $$z = z^g + \sum_{j \in A} ((\sum_{i \in R_j} p_i) - p_{r_j}) - \sum_{j \in B} ((\sum_{i \in R_j} p_i) - p_{r_j}) \tag{39}$$

   where r_j is the item whose replacement set is R_j, A is the set of replacement sets that would be selected in the process of selecting the optimal solution but were not selected in the process of selecting G (those where the algorithm has "underselected"), and B is the set of those replacement sets that were selected in the process of selecting G but would not be selected in the selection of the optimal solution (where the algorithm has "overselected").

2. In this step we consider the replacement sets that are in A. When the critical replacement set S was considered (and rejected) the set of currently selected items was exactly G. Therefore the critical replacement set was considered as the replacement for some item t that is an element of G, and was considered instead of some replacement set V that is the replacement set of some item $v \in G$ and is an ancestor replacement set of R_j. This implies that V has lower

[1] In practice the algorithm may also select other later replacement sets, replacing items in G, but since every replacement increases the total profit of the selected items (because of our assumptions in Section 6), we know that the profit of the final greedy solution is greater than or equal to z_g.

relative value (with respect to v) than S (with respect to t):

$$\frac{(\sum_{i\in V} p_i) - p_v}{(\sum_{i\in V} w_i) - w_v} \leq \frac{(\sum_{i\in S} p_i) - p_t}{(\sum_{i\in S} w_i) - w_t}.$$

Now because the replacement sets of items always have lower relative value than the replacement sets containing those items (from the hierarchical convexity assumption — see Section 6.2) all the replacement sets in A must have lower relative value than S:

$$\frac{(\sum_{i\in R_j} p_i) - p_{r_j}}{(\sum_{i\in R_j} w_i) - w_{r_j}} \leq \frac{(\sum_{i\in S} p_i) - p_t}{(\sum_{i\in S} w_i) - w_t} \quad \forall \quad j \in A. \tag{40}$$

3. In this step we consider the replacement sets that are in B. When the critical replacement set S was considered for selection (and rejected) the set of currently selected items was exactly G. There must therefore exist a list of replacement sets $J_1, J_2, J_3, \ldots, J_z$ such that $J_1 = R_j$, $J_z \subseteq G$, and J_{i+1} is the replacement set of some item in J_i for all of $i = 1, 2, 3, \ldots, z-1$.

   Likewise there also exists a list of replacement sets $M_1, M_2, M_3, \ldots, M_y$ where M_1 is the replacement set of some item in the cheapest feasible solution, $M_y = S$ and M_{i+1} the replacement set of some item m_i in M_i for all of $i = 1, 2, 3, \ldots, y-2$. Note that M_y is not selected as the replacement for some item in M_{y-1}, because M_y (ie. S) is not selected at all.

   Then we know that the algorithm at some stage replaced some item j_{z-1} in J_{z-1} with J_z instead of replacing some item m_u in M_u with M_{u+1}, for some $u \in \{1, 2, 3, \ldots, y-1\}$, since J_z was selected and S was not. Therefore

$$\frac{(\sum_{i\in J_z} p_i) - p_{j_{z-1}}}{(\sum_{i\in J_z} w_i) - w_{j_{z-1}}} \geq \frac{(\sum_{i\in M_{u+1}} p_i) - p_{m_u}}{(\sum_{i\in M_{u+1}} w_i) - w_{m_u}}.$$

   Now because the replacement sets of items always have lower relative value than the replacement sets containing those items (from the hierarchical convexity assumption — see Section 6.2) all the replacement sets in B must have greater relative value than S:

$$\frac{(\sum_{i\in R_j} p_i) - p_{r_j}}{(\sum_{i\in R_j} w_i) - w_{r_j}} \geq \frac{(\sum_{i\in S} p_i) - p_t}{(\sum_{i\in S} w_i) - w_t} \quad \forall \quad j \in B. \tag{41}$$

4. Therefore, from (39), (40) and (41) we have

$$\begin{aligned} z &\leq z^g + \sum_{j\in A}((\sum_{i\in R_j} w_i) - w_{r_j})\frac{(\sum_{i\in S} p_i) - p_t}{(\sum_{i\in S} w_i) - w_t} - \sum_{j\in B}((\sum_{i\in R_j} w_i) - w_{r_j})\frac{(\sum_{i\in S} p_i) - p_t}{(\sum_{i\in S} w_i) - w_t} \\ &\leq z^g + \left[\sum_{j\in A}((\sum_{i\in R_j} w_i) - w_{r_j}) - \sum_{j\in B}((\sum_{i\in R_j} w_i) - w_{r_j})\right]\frac{(\sum_{i\in S} p_i) - p_t}{(\sum_{i\in S} w_i) - w_t}. \end{aligned} \tag{42}$$

5. Let $\overline{c} = c - \sum_{i \in G} w_i$ be the space left in the knapsack after the selection of G, immediately before the rejection of the critical replacement set S. From the fact that S was rejected we know that the difference in cost between S and t is greater than $\overline{c}$:

$$\overline{c} \quad < \quad (\sum_{i \in S} w_i) - w_t. \tag{43}$$

Furthermore we know that the total difference in cost between the optimal solution and the intermediate greedy solution G must be less than or equal to $\overline{c}$:

$$\sum_{j \in A} ((\sum_{i \in R_j} w_i) - w_{r_j}) - \sum_{j \in B} ((\sum_{i \in R_j} w_i) - w_{r_j}) \quad \leq \quad \overline{c}.$$

6. Therefore, from (43),

$$\sum_{j \in A} ((\sum_{i \in R_j} w_i) - w_{r_j}) - \sum_{j \in B} ((\sum_{i \in R_j} w_i) - w_{r_j}) \quad \leq \quad (\sum_{i \in S} w_i) - w_t \tag{44}$$

and so, from (42) and (44),

$$\begin{aligned} z \quad &\leq \quad z^g + ((\sum_{i \in S} w_i) - w_t) \frac{(\sum_{i \in S} p_i) - p_t}{(\sum_{i \in S} w_i) - w_t} && (45) \\ &\leq \quad z^g + (\sum_{i \in S} p_i) - p_t && (46) \\ &\leq \quad z^g + \sum_{i \in S} p_i. && (47) \end{aligned}$$

Recall that the greedy algorithm compares the total profit of the final greedy solution (which is greater than or equal to z^g) to the total profit z^s of the cheapest solution containing the critical item, and keeps whichever solution is better. That is, the algorithm's solution has profit $z^h \geq \max(z^g, z^s)$. Clearly $z^s > \sum_{i \in S} p_i$, so $z^h \geq \max(z^g, \sum_{i \in S} p_i)$. Therefore, from (47),

$$z^h \quad \geq \quad \frac{1}{2} z$$

and the profit of the algorithm's solution is guaranteed to be at least half the profit of the optimal solution.

6.5 Advantages and Limitations

Recall that in Section 3.7 we noted that the maximum error of the simplified MCKP greedy algorithm was in fact bounded by the profit of the critical item, and argued that the typical behaviour of

the algorithm was therefore much better than half-optimal. Here we make a similar argument for the Hierarchical MCKP algorithm. Note that the maximum error of the greedy algorithm is bounded by the difference in profit between the critical replacement set and the item it replaces (see equation (46)). Therefore as the granularity of the candidate items with respect to the knapsack becomes finer, the maximum error of the algorithm tends to zero. In practical level of detail applications the performance of the algorithm can be expected to be much better than half-optimal. Pathological cases arise only when the difference in total profit between S and t is a significant proportion of the optimal solution value z.

Recall that just as its ancestor the simplified greedy algorithm for MCKP described in Chapter 3 depends on the non-hierarchical convexity assumption, the algorithm depends on the hierarchical convexity assumption (Section 6.2) for its half-optimality. Its solution to instances of the Hierarchical MCKP that do not satisfy the hierarchical convexity assumption may be less than half-optimal in the worst case. The hierarchical convexity assumption implies that more expensive selections must provide increased profit at the expense of increased cost, with diminishing returns for increasingly more expensive selections. These requirements are likely to be satisfied in most real-world applications. Level of detail problems in which more expensive renderings do not provide diminishing returns are uncommon. For example the successive addition of more detail to a polygonal mesh model generally results in progressively smaller improvements in visual perception. Therefore the half-optimality guarantee is a worst case figure and the algorithm can be expected to perform well in most practical level of detail applications.

The complexity of the greedy algorithm is $O(n \log n)$. The number of replacements made is $O(n)$ in the number of candidate items (and hence in the number of replacement sets). Each replacement involves the deletion of the replacement set at the head of the list, which is $O(1)$, and the in-order insertion of (we assume) $O(1)$ new replacement sets, each at the expense of $O(\log n)$ complexity. The complexity of the greedy selection stage is therefore $O(n \log n)$. After the greedy selection stage the critical item solution must be found. The complexity of this is $O(n)$. The entire algorithm is therefore $O(n \log n)$.

6.6 Incremental Version

Like the simplified greedy algorithms for the MCKP presented in Chapter 3, the hierarchical MCKP algorithm can be made incremental to take advantage of coherence between the optimal solutions of successive problem instances. That is, it can be modified to take as input an initial approximate

solution that is typically derived from the most recent application of the algorithm. The fact that the algorithm can be made incremental is a direct result of the hierarchical convexity assumption. We provide an incremental hierarchical level of detail optimization algorithm based on this algorithm in Chapter 7. That incremental algorithm is equivalent to this greedy algorithm for instances of the Hierarchical MCKP for which the hierarchical convexity assumption defined in Section 6.1 holds.

6.7 Summary

In this chapter we have presented a greedy approximation algorithm for the Hierarchical Multiple Choice Knapsack Problem introduced in Chapter 4. Our algorithm is half-optimal for a useful subproblem of the Hierarchical MCKP and has a time complexity of $O(n \log n)$. Furthermore its worst case performance is significantly better than half-optimal for instances of the subproblem in which the profits of the candidate items are small with respect to the total profit of the optimal solution.

The algorithm may be made incremental to take advantage of coherence between the optimal solutions of successive problem instances, and we will make use of this in the development of an incremental hierarchical level of detail optimization algorithm in the next chapter, Chapter 7.

Chapter 7

Hierarchical Level of Detail Optimization Algorithm

In this chapter we present our level of detail optimization algorithm. Our algorithm is *incremental, hierarchical* and *predictive*. It is an incremental version of the greedy approximation algorithm for the Hierarchical Multiple Choice Knapsack Problem presented in Chapter 6, rewritten to exploit frame-to-frame coherence for increased efficiency in practical level of detail applications. This basis in an approximation algorithm for the Hierarchical MCKP gives the algorithm its truly hierarchical nature with elegant support for multiple shared simplified representations for groups of related objects. In addition it implies that the algorithm is predictive and provides both guaranteed limits on predicted rendering cost and guaranteed levels of predicted visual quality.

This incremental algorithm exploits frame-to-frame coherence by accepting as input an initial solution that is the approximate solution found for the previous frame. Due to the typically large amount of coherence between successive frames, the approximate solutions found for consecutive frames are generally similar. Therefore the approximate solution is found more efficiently on average by the incremental algorithm than by the non-incremental greedy algorithm. In this way our algorithm is similar to the incremental non-hierarchical level of detail optimization algorithm of Funkhouser and Séquin (discussed in Section 2.6.1). It is equivalent to the greedy algorithm for the Hierarchical MCKP presented in Chapter 6 for a useful subproblem. This is, by coincidence, the same subproblem for which the greedy algorithm's solution is at least half-optimal. Therefore the algorithm produces an approximate solution that is guaranteed at least half as good as the optimal solution, for the subproblem of the hierarchical level of detail optimization problem in which higher levels of detail provide increased perceptual benefit but with diminishing returns. We prove the

equivalence of the algorithms for this subproblem formally in Section 7.2.

The incremental algorithm shares the advantages of the non-incremental algorithm of Chapter 6. It has a worst-case theoretical time complexity of $O(n \log n)$, but is incremental and so has an average time complexity close to $O(n)$.[1] The algorithm produces an approximate solution that is guaranteed at least half as good as the optimal solution, for the subproblem of the hierarchical level of detail optimization problem in which higher levels of detail provide increased perceptual benefit but with diminishing returns.

Whereas the greedy algorithm of Chapter 6 was described in terms of the Hierarchical Multiple Choice Knapsack Algorithm (that is, *candidate items* with profit and cost, partitioned into a number of *candidate subsets*) this incremental algorithm will be described in terms of hierarchical level of detail optimization (that is, *impostors* with perceptual benefit and rendering cost, each belonging to a particular *scene object*).

The definition of efficient and accurate benefit and cost heuristics for use in the algorithm is itself a tricky problem. However we are more interested in level of detail optimization algorithms than in the problem of defining the heuristics themselves and so we assume that appropriate *benefit* and *cost* heuristics exist that predict the perceptual benefit and rendering cost respectively of impostors in any given viewing situation (see Chapter 2). In Chapter 9 we report on an experiment in which some simple heuristics were evaluated.

The organization of this chapter is as follows. In Section 7.1 we present the algorithm. In Section 7.2 we prove the equivalence of the incremental algorithm to the greedy algorithm for the Hierarchical MCKP described in Chapter 6. In Section 7.3 we discuss the advantages and limitations of the algorithm. In Section 7.4 we conclude the chapter with a summary of the important points.

7.1 Algorithm

The incremental hierarchical level of detail optimization algorithm, shown in Figure 41, is an equivalent incremental version of the Hierarchical MCKP greedy algorithm described previously in Chapter 6. Its advantage over that algorithm is purely one of efficiency: it exploits coherence by basing its initial solution on the solution reached for the previous frame.

The algorithm is applied once per frame and its output is a level of detail of the scene object for that frame. Its input is the level of detail selected for the previous frame[2] and a constant *rendering*

[1] A detailed experimental evaluation of its practical complexity is presented in Chapter 9.

[2] Or any valid level of detail, in the case of the first frame.

input: an instance of the hierarchical level of detail optimization problem
input: a feasible initial solution to that instance
output: a solution to that instance (in L)

begin
    set $L \leftarrow$ the initial solution
    set done $\leftarrow$ FALSE
    while done = FALSE
    {

        // increment L, if possible

        if L is not the highest level of detail **then**
        {
            find i, the impostor in L whose replacement set has highest relative value
            set $R \leftarrow$ the replacement set of i
            set $L \leftarrow (L - \{i\}) \cup R$
        }

        // decrement L, while the total rendering cost is too high

        while $\sum_{i \in L}$ Cost(i) $>$ *rendering cost limit*
        {
            find $S \subseteq L$, the replacement set in L with the lowest relative value
            set $j \leftarrow$ the impostor whose replacement set is S
            set $L \leftarrow (L - S) \cup \{j\}$
            if $S = R$ **then**
                set done $\leftarrow$ TRUE
        }
    }
end

Figure 41: **The incremental hierarchical level of detail optimization algorithm.**

cost limit that represents the rendering time available for this frame. The algorithm guarantees that the total predicted rendering cost of the selected level of detail is lower than the rendering cost limit. In addition it attempts to maximize the total predicted perceptual benefit (or profit) of the selection.

The algorithm is iterative, repeatedly incrementing and decrementing the selected level of detail until the final solution is found. In each iteration the selected level of detail is incremented once, then decremented repeatedly while the total rendering cost is greater than the rendering cost limit. Recall that a level of detail may in general be incremented and decremented in many different ways. The incrementation selected in each step is that which replaces the currently selected impostor whose *replacement set* has the *highest relative value*. Conversely the decrementation selected is that which deselects the *currently completely selected* replacement set with the *lowest relative value*.

The repeated iteration terminates when the incrementation and decrementation operations of the same iteration select and deselect the same replacement set. When this occurs there is no further work for the algorithm to do. After termination of the algorithm we render the impostors constituting the selected level of detail. For simplicity the algorithm as shown assumes that the rendering cost limit is sufficient to render at least the lowest level of detail but not great enough to render the highest level of detail. The special cases in which these assumptions do not hold must be dealt with by trivial tests before the algorithm begins. In such cases we select the lowest and highest levels of detail, respectively.

Note also that the algorithm as described here does not consider the *critical replacement set solution* (see Chapter 6), the cheapest feasible solution containing the *critical replacement set*. For completeness the algorithm should, after termination of the iterative stage, compare the solution reached against the critical replacement set solution and take whichever has greater total profit. In the case where the iteration terminates due to the selection and deselection of the same replacement set in a single iteration, the critical replacement set is the replacement set in question (since the incremental algorithm is equivalent to the non-incremental one — as we show in Section 7.2 — so all the replacement sets selected have higher relative value than it and all those not selected have lower relative value than it). In the cases where the algorithm terminates due to running out of possible decrementations or incrementations, the critical replacement set is either the first replacement set or does not exist, respectively. Once the critical replacement set is known the critical replacement set solution can be found using the algorithm presented previously in Figure 40 (Section 6.3).

In practice we believe that checking the critical replacement set solution is unnecessary for level of detail optimization. Recall that the critical replacement set solution is the lowest cost feasible

solution containing the critical replacement set. Therefore in any instances in which the critical replacement set solution has a higher total profit than the solution resulting from greedy selection, the critical replacement set itself must have a profit that is large relative to the total profit of the optimal solution. Such instances seem unlikely in level of detail optimization where the profit of the individual impostors is likely to be small relative to the total profit of the optimal solution.

7.2 Equivalence of the Incremental and Non-Incremental Algorithms

In this section we prove that the incremental hierarchical level of detail optimization algorithm described in this chapter is equivalent to the non-incremental greedy algorithm for the Hierarchical Multiple Choice Knapsack Problem described in Chapter 6. We do this by using level of detail graph representations (Chapter 5) of the state spaces of the incremental and non-incremental algorithms to analyze their behaviour and show that their solution states are always identical.

Both algorithms, as we noted previously, are guaranteed to be half-optimal for well-defined subproblems of their respective problems (the hierarchical level of detail optimization problem and the Hierarchical MCKP). In addition they are only equivalent for these subproblems: if the requirements of those subproblems are violated then the incremental algorithm is not equivalent to the non-incremental algorithm and its output may differ from that of the non-incremental one. We hence assume here that the requirements are met: that the replacement set of any impostor (or item) has higher profit and cost than that impostor and lower relative value than the replacement set (if any) containing that impostor.

In this proof we refer to the concepts of *ancestor* and *descendant* replacement sets, the *covering* of replacement sets by levels of detail, and the *incrementation* and *decrementation* of one level of detail to another. These concepts were introduced in Chapter 4.

7.2.1 Level of Detail Optimization as a Search Problem

Recall from Chapter 5 that each instance of a hierarchical level of detail description generates its own particular level of detail graph. The hierarchical level of detail optimization problem may be viewed as a search problem involving the traversal of the level of detail graph (from some initial level of detail) in search of an optimal level of detail in which total profit is maximized while total cost is limited. The operation of both the greedy algorithm for the Hierarchical MCKP and the incremental hierarchical level of detail optimization algorithm may in turn be viewed as different traversals in search of a solution state that is an approximation to this optimal state. Our aim in this

proof is to show that both algorithms reach the same solution state.

The greedy algorithm for the Hierarchical MCKP presented in Chapter 6 begins with the lowest level of detail selected (that is, the lowest cost solution) and always increments from one level of detail to another towards its final solution by replacing selected items with their replacement sets. The currently selected item (or impostor) chosen for replacement in each step, we recall, is that whose replacement set has *greatest relative value*. If the replacement can't be afforded then that replacement set is simply discarded. The greedy algorithm stops when it runs out of available replacement sets – when it reaches a level of detail in which no further affordable replacements exist. That level of detail is then its final solution.

The incremental hierarchical algorithm on the other hand begins with any arbitrary level of detail selected (usually, the approximate solution found for the previous frame) and both increments and decrements to reach its final solution. Each incrementation is the replacement of a currently selected impostor with its replacement set and each decrementation is the replacement of a currently selected replacement set with its associated impostor. In each iteration the impostor selected for replacement during incrementation is that whose replacement set has *greatest relative value* and the replacement sets selected for replacement during decrementation are those completely selected replacement sets with *lowest relative value*. The incremental algorithm terminates when it reaches a level of detail in which the selected incrementation is negated in the same iteration by its inverse decrementation.

The nodes of the level of detail graph are partitioned into two subsets 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 replacement sets always have higher cost than the items they replace, higher levels of detail always have higher cost, so we can imagine the level of detail graph being cut into two distinct self-contained parts by an imaginary surface. The optimal and approximate solutions must all lie just below this surface. Figure 42 compares the actions of the incremental algorithm and non-incremental greedy algorithm as traversals of a level of detail graph with respect to this surface.

Let the solution state found by the non-incremental greedy algorithm be called g. When the incremental hierarchical 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 or replacement set. In general, some of the incrementations will select replacement sets that are *covered* (See Definition 4.5 in Section 4.1.5) by g but not by s, and others will select replacement sets that are covered by neither s nor g. Similarly decrementations will generally be available that deselect replacement sets that are covered by s but not by g and others that deselect replacement sets that are covered by both s and g. The central idea behind the

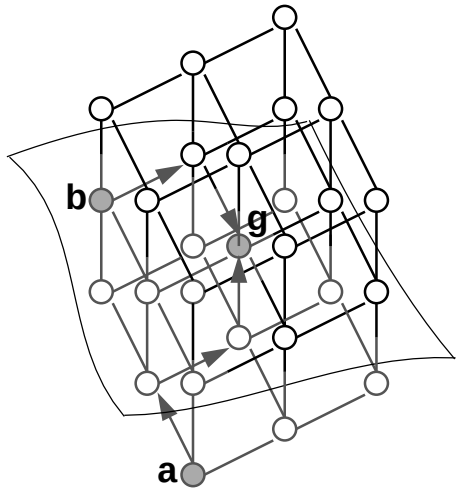


Figure 42: **Level of detail optimization as a search problem.** The non-incremental greedy algorithm begins at the lowest level of detail (a) and increments until it reaches a final solution (g). The incremental algorithm begins at the previous frame's solution (b) and both increments and decrements to reach (g). Also shown is the imaginary cutting surface between the levels of detail whose cost is less than or equal to the available rendering time (feasible solutions, shown with grey lines) and those whose cost is not (infeasible solutions, shown with black lines).

proof is to show that the incrementations and decrementations chosen always serve to bring the selected state s "closer" to g, by selecting whenever possible replacement sets that are covered by g and deselecting ones that are not. By showing that the incremental algorithm terminates only upon reaching g, we prove that the solution state of the incremental algorithm is also g.

7.2.2 Incrementation and Decrementation

In this section we prove two lemmas that together characterize the behaviour of the incremental algorithm with respect to the final solution state g of the non-incremental algorithm, 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 43). 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 $s \neq g$).

The impostors and replacement sets referred to in the proofs of the lemmas are illustrated in a simple example in Figure 43. This example consists for simplicity of the level of detail graph of a simple non-hierarchical level of detail description with two objects. Note however that the same arguments apply for complex level of detail graphs of arbitrary hierarchical level of detail descriptions.

Lemma 7.1 *Whatever its current state, the incremental algorithm 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.*

Proof:

1. *In the case where the current state t of the incremental algorithm is in the class $s < g$, all possible incrementations select replacement sets covered by g and the proof is immediate.*

2. *In the case where t is in class $s = g$ or $s > g$ all incrementations select replacement sets covered neither by t nor by g, and the proof is immediate.*

3. *In the case t is in class $s \neq g$, there must exist at least one incrementation from t that selects a replacement set i covered by g. At least one incrementation must select a replacement set j not covered by g (by definition of this class). There must exist a state t' from which the non-incremental greedy algorithm chose an incrementation selecting i over one selecting another replacement set j' that is an ancestor replacement set of j. Replacement set i must therefore have greater relative value than j'. Since descendant replacement sets have lower relative value than their ancestors, j' has greater relative value than j and so i has greater relative value than j. Therefore the incremental algorithm will choose the incrementation selecting i over that selecting j.*

Lemma 7.2 *Whatever its current state, the incremental algorithm will always choose a decrementation deselecting a replacement set not covered by g, if one is available, over any that deselect replacement sets that are covered by g.*

Proof:

1. *In the case where the current state u of the algorithm is uniformly higher than g, any decrementation must deselect a replacement set not covered by g, and the proof is immediate.*

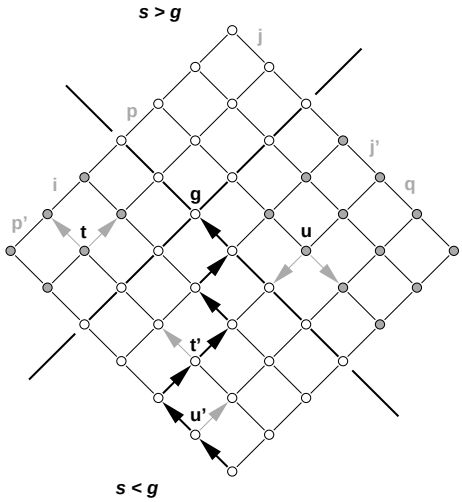


Figure 43: **Partitioning of states.** The partitioning of the set of states into four distinct classes: $s = g$, $s > g$, $s < g$ and $s \neq g$ (the shaded states), for a regular 2D level of detail graph. The dark arrows show, for this example, the path taken by the non-incremental algorithm in reaching its final solution state g. The light labels on rows of arcs name the replacement sets selected and deselected by all the incrementations and decrementations represented by arcs in that row, and the light arrows to the possible incrementations and decrementations mentioned in the text.

2. *In the cases where u is in class $s < g$ or $s = g$, no decrementations exist that deselect replacement sets 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 a replacement set p not covered by g. At least one decrementation from u must deselect a replacement set q that is covered by g (by definition of this class). There must exist a state u' from which the non-incremental greedy algorithm chose an incrementation selecting q over one selecting another replacement set p' that is an ancestor replacement set of p. Replacement set q must therefore have greater relative value than p'. Since descendant replacement sets have lower relative value than their ancestors, p' has greater relative value than p, so q has greater relative value than p. Therefore the incremental algorithm will choose the decrementation deselecting p over that deselecting q.*

Together these lemmas 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.

7.2.3 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 a replacement set 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 rendering cost 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.

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

Table 1: **Actions of the incremental level of detail algorithm.** 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 replacement set selected is covered by g, the number of decrementations performed, whether or not the replacement set(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 a replacement set 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 still $s < g$.

In class $s \neq g$ the incremention also selects a replacement set 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 may not be greater than the limit, and so may or may not decrement. Any decrementations that are performed will

deselect replacement sets 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 44. 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 indefinitely and must eventually move to class $s = g$ and terminate. Its final solution is therefore always g, the non-incremental algorithm's solution.

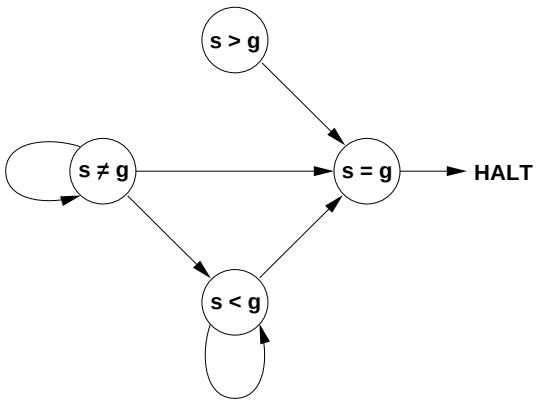


Figure 44: **Summarized state diagram of the incremental algorithm.** The four state classes (represented by circles) correspond to the four possible relationships that the algorithm's current state (or level of detail) s can have to the solution state of the non-incremental algorithm, g. In each iteration of the incremental algorithm it changes its current state, either moving from one state class to another or staying in the same class.

In this proof we have assumed that the incremental algorithm terminates by means of its normal terminating case: that it selects (by incrementation) and deselects (by decrementation) the same replacement set in a single iteration. This assumes that the exceptions in which the rendering cost limit is either too low to select any representation or high enough to select the most expensive representation have been eliminated by means of simple checks as described in Section 7.1.

We have also ignored the case in which the algorithms select the critical replacement set solution: the lowest cost feasible solution containing the critical replacement set. In this case both algorithms select it (assuming that the test of the critical replacement set solution is not skipped in the incremental algorithm) and their solutions are the same. As we noted in Section 7.1, the critical replacement set solution is unlikely to be useful in practice since the algorithm is intended for level

of detail optimization.

7.2.4 Proof for Funkhouser-Séquin Algorithm

Note that the proof of equivalence presented in Section 7.2 may be altered trivially to act as a proof of equivalence of the incremental and non-incremental versions of Funkhouser and Séquin's algorithm (Sections 2.5.4 and 2.6.1). This follows from the fact that the relationship between our incremental and non-incremental algorithms is essentially the same as the relationship between theirs. The differences are that the Funkhouser-Séquin algorithm is non-hierarchical and deals with *value* rather than *relative value*. The proof may therefore be adapted by replacing all references to *replacement sets* with references to *replacement items* and all references to *relative value* with references to *value*. In addition the concepts of *descendant* and *ancestor* replacement sets, as well as the *covering* of replacement sets by levels of detail, must be used in their simpler non-hierarchical form.

7.3 Advantages and Limitations

The complexity of the incremental algorithm is $O(n \log n)$ in the worst case. One possible implementation might make use of two lists of replacement sets, ordered by increasing and decreasing relative value. One list contains candidates for selection during incrementation and the other candidates for deselection during decrementation. Then the tasks of finding the currently selected impostor whose replacement set has greatest relative value and the currently completely selected replacement set with lowest relative value are reduced in complexity to $O(1)$. In exchange, the ordered lists must be updated after every incrementation and decrementation by means of the deletion of the replaced impostors and the insertion, in order, of the newly selected replacement sets. This can be performed optimally with a complexity of $O(\log n)$. The number of decrementations performed per iteration is $O(1)$ and the number of iterations in the worst case is $O(n)$. Therefore the theoretical worst-case complexity of the algorithm is $O(n \log n)$ where n is the number of object impostors.

We have found in practice that it may be difficult to find an implementation that allows the theoretical complexity of $O(n \log n)$. In particular, our experimental implementation presented in Chapter 9 has a worst-case time complexity of $O(n^2)$. The stated $O(\log n)$ complexity of the updates to the incrementation and decrementation lists depends on the lists being stored in some optimal data structure such as a tree. If a tree is used however it will require constant rebalancing

to prevent it degenerating into a list, since replacement sets are always removed from the front. In our experimental implementation (described in Chapter 9) we made use of unsorted arrays rather than more complex data structures, thereby incurring the cost of searching the arrays before every incrementation and decrementation, but avoiding the costs of updating the lists afterwards (by inserting items in order). Moreover, this allows us to use arrays rather than lists, leading to increased benefits due to caching. In this sense the complexity arguments are useful as guidelines only and what matters in practice is the efficiency of the implementation.

In any case only a few iterations are generally performed for each frame, due to the typical coherence between the approximate solutions of successive frames. Rather than beginning with the lowest level of detail and repeatedly incrementing to reach the final solution as the non-incremental algorithm of Chapter 6 does, this algorithm begins with the final solution from the previous frame (which is typically similar to the final solution for this frame) and both increments and decrements to reach the final solution. The actual number of incrementations and decrementations performed varies depending on frame-to-frame coherence. In typical situations where frame-to-frame coherence is high, the number of iterations (and therefore the practical complexity of the algorithm itself) approaches $O(1)$. Our experimental results in Chapter 9 suggest that its average complexity is close to $O(n)$.

Situations where the complexity approaches $O(n \log n)$ or $O(n^2)$ include the first frame, in which the initial solution is simply an arbitrarily chosen level of detail (typically the lowest), and those where the visual importance of many objects changes dramatically from one frame to the next. In these cases the initial and final solutions may be quite dissimilar and a significant number of incrementations and/or decrementations must be performed. In the worst case the efficiency of the incremental algorithm approaches that of the non-incremental one. We report on an experimental investigation of the algorithm's efficiency in Chapter 9.

The incremental algorithm represents a hierarchical generalization of the optimization approach of Funkhouser and Séquin (plus some adaptations to ensure half-optimality) at little or no expense in terms of time complexity. The time complexities of the optimization algorithms of Funkhouser and Séquin and Maciel and Shirley, by comparison, are $O(n \log n)$ and $O(n)$ respectively[3]. The algorithm of Maciel and Shirley however is not incremental and therefore requires a complete greedy selection process starting from the lowest level of detail for every frame.

Our optimization algorithm has several important advantages over both the non-hierarchical

[3]Maciel and Shirley claim in [47] that the complexity of their algorithm may be reduced to $O(\log n)$, but we see no way of doing this – see Chapter 2.

algorithm of Funkhouser and Séquin and the hierarchical algorithm of Maciel and Shirley. Firstly, it is hierarchical and allows the use of hierarchical level of detail descriptions of arbitrary structure and type with multiple shared representations for groups of objects. This enables the algorithm to save additional rendering cost by selecting shared simple representations, or impostors, for groups of less important objects and to guarantee a complete scene representation even when the complexity of the visible scene is very high. This affords better renderings of more important objects and prevents the appearance of "holes" as objects are omitted completely in order to limit rendering cost. In this way the algorithm is an improvement over that of Funkhouser and Séquin [24] and is similar to that of Maciel and Shirley [47].

Secondly, the algorithm provides a solution that is guaranteed to be at least half as good as the optimal solution for a restricted subproblem of the hierarchical level of detail optimization problem in which higher levels of detail are more expensive and provide diminishing returns. In this way it is an improvement over the algorithms of both Funkhouser and Séquin and Maciel and Shirley, whose algorithms provide no such guarantee (See Sections 2.6.1 and 4.3). Furthermore, as we showed for our greedy algorithms for the MCKP in Section 3.7 and the Hierarchical MCKP in Section 6.5, the algorithm's solution is typically much better than half-optimal as long as the granularity of the impostors is relatively small (that is, as long as there are no impostors whose individual benefits contribute a significant proportion of the total benefit of the optimal solution).

The incremental hierarchical algorithm is *predictive* (See Section 2.2) in that it actively attempts to regulate the rendering complexity of the scene representations selected in each frame. The predicted rendering time of its solution is always less than or equal to the available frame time. It, like the algorithm of Funkhouser and Séquin, is therefore better able to guarantee interactivity in the form of constant rendering times than are the non-predictive algorithms of for example Shade *et al* [75] and Chamberlain *et al* [14].

The most significant limitation of the algorithm results from the assumption that the replacement sets of impostors always have greater rendering cost and profit than those impostors and lower relative value than the replacement sets (if any) containing those impostors on which the equivalence to the non-incremental algorithm is based. If this requirement is not met then the incremental and non-incremental algorithms are no longer equivalent and no guarantees can be placed on the quality of the solution.

This *diminishing returns* assumption is identical to the *hierarchical convexity assumption* defining the subproblem of the Hierarchical MCKP for which the solution of the greedy algorithm of Chapter 6 is at least half-optimal. In Section 6.5 we argued that this requirement was not excessively

restrictive. Similar arguments apply here. The diminishing returns restriction agrees well with the common conception of higher levels of detail as *more expensive* and *less efficient* representations to be used only when they can be afforded. Moreover, we believe diminishing returns to be commonplace in rendering; gains in perceptual benefit due to more complex renderings seldom match their accompanying increases in rendering cost. We expect, for example, that successive increases in the polygon-count of an object model will tend to result in progressively smaller improvements in the perception of the modeled object. For these reasons, we believe that the diminishing returns requirement does not significantly impair the algorithm's usefulness.

At any rate, the requirements are ultimately restrictions on the benefit and cost *heuristics*, rather than on the actual perceptual benefit and rendering cost of object representations. For example, fluctuations in the predicted perceptual benefit and rendering cost of impostors due to viewing orientation must be carefully controlled. Impostors most likely to cause problems are those whose perceptual benefit or rendering cost are strongly dependent on the angle from which they are viewed – for example, single large textured polygons. In particular, problems are likely to arise in cases where different impostors of the same object are affected differently by changes in viewing direction: in these cases the ordering of the impostors may change as viewing direction changes. In the worst case, orientation dependent effects can simply be ignored by making benefit and cost heuristics independent of viewing direction at the expense of some reduction in accuracy. Care must be taken to ensure that the requirements are satisfied in all conceivable rendering situations.

Notably the algorithm of Funkhouser and Séquin (Chapter 2) also has a similar (but non-hierarchical) limitation. Their incremental level of detail algorithm is only equivalent to their MCKP greedy algorithm if higher levels of detail of objects always have higher profit and cost and lower *value*. The manner in which their decreasing value assumption allows their incremental algorithm to be equivalent to their non-incremental algorithm corresponds directly to the manner in which our decreasing relative value assumption allows our incremental and non-incremental algorithms to be equivalent.

7.4 Summary

In this chapter we have presented an incremental hierarchical level of detail optimization algorithm. This algorithm is based on the equivalence of the hierarchical level of detail optimization problem to a hierarchical version of the Multiple Choice Knapsack Problem, noted in Chapter 4, and is an incremental version of the greedy algorithm for that Hierarchical MCKP presented in Chapter 6.

The advantage of this algorithm over the simple greedy algorithm is that it accepts as input an initial "best-guess" solution that is the approximate solution found for the previous frame and so is able to exploit the typically large coherence between successive frames in level of detail optimization.

We have also shown that the incremental hierarchical algorithm is equivalent to the greedy algorithm and provides an approximate solution that is guaranteed to be at least half-optimal for a restricted subproblem of the hierarchical level of detail optimization problem. The assumption defining the subproblem is that replacement sets of impostors should always have greater total profit and cost than the impostors they replace, and lower *relative value* than the replacement sets (if any) containing those impostors. The implications of this in terms of level of detail is that higher levels of detail of objects must always provide diminishing returns.

We used level of detail graphs (introduced in Chapter 5) to prove the equivalence of the incremental and non-incremental versions of the algorithm. Central to this proof is the realization that level of detail optimization may be regarded as a search problem on a level of detail graph.

The next chapter presents the findings of an experiment designed to demonstrate the practical usefulness of hierarchical level of detail optimization.

Chapter 8

Perceptual Experiment

This chapter describes a perceptual experiment that was conducted in order to compare the use of hierarchical level of detail optimization to traditional non-hierarchical level of detail optimization. In particular, we wish to determine the effects of the use of hierarchical level of detail descriptions with shared representations for groups of objects that our hierarchical algorithm allows.

In order to compare the use of hierarchical level of detail descriptions to the use of non-hierarchical ones, we needed a predictive non-hierarchical level of detail optimization algorithm. We selected the incremental algorithm of Funkhouser and Séquin [24] (described in Chapter 2), since it is the best predictive non-hierarchical algorithm that we are aware of, and because our hierarchical algorithm is closely related to it. This experiment serves as a demonstration of the effectiveness of extending predictive level of detail optimization to make use of hierarchical level of detail descriptions, and is therefore also in a sense an evaluation of the general approach used by Maciel and Shirley [47].

For the purposes of this experiment we introduce the use of *perceptual evaluation* for the practical investigation of graphics algorithms. In keeping with the *user-centric* approach of level of detail, perceptual evaluation involves the subjective controlled evaluation of rendered images by a group of volunteer non-expert users. We derive much of our inspiration and methodology in this regard from the more established practice of the perceptual evaluation of television pictures, as described in [12].

We begin the chapter in Section 8.1 by discussing the experimental hypothesis that the experiment is designed to test. In Section 8.2 we describe the experimental methodology used. In Section 8.3 we present and discuss the results of the experiment. We close the chapter in Section 8.4 with some concluding remarks.

8.1 Aims

As was discussed in Chapter 7, the principle advantage of the use of impostors for group objects is that the level of detail optimization algorithm is able to select these group impostors in situations where the rendering cost incurred by rendering even the cheapest representations of the individual group components would be better invested in the improved rendering of other objects. Due to the inherent complexity of multiple disjoint representations, it is generally possible to provide shared representations that are simpler and therefore cheaper to render than even the simplest reasonable representations of the individual objects. This allows the optimization algorithm to better cope with arbitrary complexity of the visible scene by selecting, where appropriate, progressively less detailed impostors for progressively larger groups of objects. In particular, it enables the algorithm to fulfill the maximum rendering cost requirement in situations where the maximum permitted rendering cost is lower than the *nominal cost* of the scene, without creating "holes" in the rendered images by omitting scene objects entirely through the selection of *null* impostors (impostors with no drawable representation).

We define the *nominal cost* of a scene in a particular situation to be the minimum cost of rendering the scene in that situation without the use of null or group impostors. That is, the total cost of the lowest non-null impostors of the visible leaf objects.

We hypothesize that the use of hierarchical level of detail descriptions with shared group object representations is capable of improving the effectiveness of predictive level of detail optimization: that the perceptual benefit of rendering may be improved for no increase in rendering cost. We claim that the provision of shared group object representations and the ability to select them in level of detail optimization allows the selection of a complete scene representation for every frame, avoiding the appearance of holes. Furthermore we claim that hierarchical level of detail descriptions allow for the saving of rendering cost that may be used to improve the rendering of more important objects.

8.2 Methodology

In this section we describe in detail the methodology employed in the evaluation. We discuss the the evaluation scheme used, the content and selection of the image sequences used, the selection of assessors, the experimental conditions, the evaluation scheme, the experiment procedure, and the details of the optimization algorithms compared.

8.2.1 Approach

The approach taken for the experimental evaluation was one of *perceptual evaluation*, which is more commonly used in the evaluation of the perceptual image quality of television equipment [81] [31] [66] [12]. The general approach of performing user studies has also been used recently by Smets and Overbeeke [79], Watson *et al* [84] [85] and Reddy [63] [62].

Perceptual evaluation involves the subjective perceptual evaluation of a series of image sequences by a group of typical human assessors. We selected the *stimulus comparison* method [12], in which image sequences produced using two competing methods are compared in pairs and an index of the relationship between the two sequences of each pair is provided by each assessor. This method gives rise to 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.

The level of detail optimization algorithms used to produce the image sequences to be compared were (an early version of) the incremental hierarchical level of detail optimization algorithm of Chapter 7 (for hierarchical optimization) and the non-hierarchical level of detail optimization algorithm of Funkhouser and Séquin (for non-hierarchical optimization). The behaviour of the hierarchical and non-hierarchical algorithms becomes more as the *rendering cost limit* (the maximum permitted rendering cost of each frame) increases. Conversely as the rendering cost limit decreases the hierarchical algorithm is more likely to select simple shared representations for group objects in order to satisfy it, so that the hierarchical and non-hierarchical results become less similar. We therefore concentrated on very low rendering cost limits.

8.2.2 Image Content

Our test scene consists of a group of geodesic domes, where each dome is constructed from a set of cylinders. Figure 45 shows two images of the scene. Cylinders were judged to be suitable leaf objects due to the ease with which multiple impostor representations of them at varying levels of detail can be created. Domes were selected as group objects due to the availability of suitable group impostors in the form of low-resolution spheres.

The entire scene consists of 16 animated geodesic domes, arranged pseudo-randomly in the x and y directions so as to partially fill the field of view. Their z (distance) coordinates are calculated in each frame according to a sine-based function of the elapsed time (number of frames), so that each dome moves periodically towards and away from the viewer. The phase of the function for each

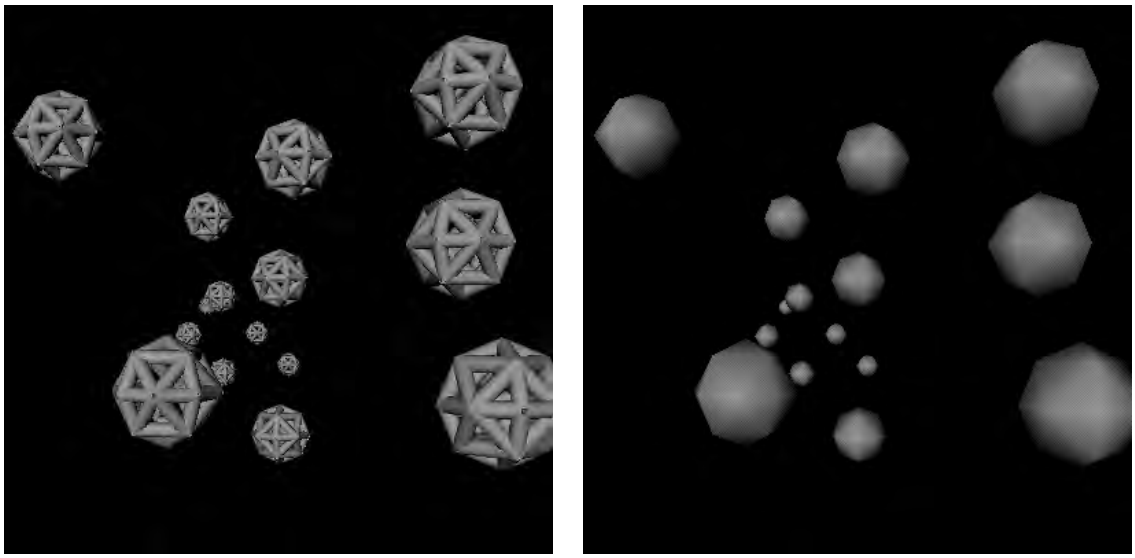


Figure 45: **Scene used in the perceptual experiment.** The scene consists of 16 geodesic dome group objects, each constructed from 48 cylindrical parts. The domes are represented implicitly in the image on the left by the explicit cylinder representations of their cylinder parts. In the image on the right, they are represented explicitly by their associated spherical impostor representations.

dome is chosen pseudo-randomly, so that the domes are out of step. This allows a wide distribution of distances from the viewer, while ensuring that each dome object is at times relatively close by and at other times relatively far off. The level of detail algorithms are therefore forced to continually and dynamically update the levels of detail of all objects.

Each geodesic dome used in the experiment consists of 48 cylinders. The use of impostors for group objects in the hierarchical case is facilitated by this hierarchical structure of larger objects composed of many smaller parts.

While this content does not represent worst case content for the effects under investigation, it does serve to test the effects under circumstances when they are likely to be significant. The usefulness of impostors for general group objects can not be extrapolated entirely from this experiment, since it will depend in general on the availability of group impostors with similar perceptions to that of the implicit representations of those groups. We expect however that typical scenes will contain many cases in which impostors for group objects may be employed as usefully as they are here.

8.2.3 Stimuli

In this section we describe in detail the stimulus material selected for use in the experiment.

Rendering Cost Limits

Table 2 shows the range of rendering cost limits used in the experiment.

Rendering cost limit	Description
1920	Half of the nominal cost
3840	Equal to the nominal cost
7680	Twice the nominal cost
15360	Equal to the sufficient cost

Table 2: **Rendering cost limits.** The range of rendering cost limits used in the experiment.

The nominal cost (Section 8.1) of rendering all objects at their lowest non-null levels of detail without the use of impostors for group objects is equal to the sum of the rendering costs of the lowest detail non-null impostors of all the leaf objects. In this case, therefore, it is equal to the number of domes (16) multiplied by the number of cylinder objects per dome (48), multiplied by the Cost of the lowest detail non-null level of detail of each cylinder object. Since the rendering cost of each object representation in this implementation is assumed to be view-independent, the cost of the lowest detail cylinder impostor LoD_1 is always equal to 5. In addition, all objects are always within the viewport and we make no attempt to take into account the effects on rendering cost of occlusion of one object by another. Therefore, the nominal cost of the scene is always equal to 3840.

One aim of the experiment is to compare the behaviour of the two algorithms under critical conditions, where there is insufficient rendering time available to render all objects at their lowest level of detail. That is, when the rendering cost limit is less than the nominal cost. In practice, when either group or null impostors are available, the hierarchical and non-hierarchical algorithms respectively will choose to select these impostors even for some situations where the rendering cost limit is close to but greater than the nominal cost. This occurs because the use of null or group impostors for less important objects represents a saving of rendering cost that may be better utilized in the improved rendering of more important objects, as with any other low levels of detail. By providing group or null impostors at all, we invite their selection even when not absolutely

necessary.

For this reason, we selected a range of rendering cost limits encompassing a figure equal to the nominal cost, one with a figure equal to half of the nominal cost, and one with a figure twice that of the nominal cost. In addition, a single image sequence was created at a rendering cost limit equal to the rendering cost of the highest detail representation of the scene, for use as a reference. This full-detail sequence was rendered with the non-hierarchical algorithm, but could have been rendered with either algorithm, since they behave identically when the rendering cost limit is equal to the maximum rendering cost.

Image Sequences

Rendering cost limits were selected as described in Section 8.2.3. Table 3 shows the image sequences created, and the parameters of each.

Sequence	Algorithm	Rendering cost limit
1	hierarchical	7680
2	non-hierarchical	7680
3	hierarchical	3840
4	non-hierarchical	3840
5	hierarchical	1920
6	non-hierarchical	1920
7	non-hierarchical	15360

Table 3: **Parameters.** Parameters of the image sequences selected for the experiment.

Figures 46, 48, 49 and 47 show the first frame from each of the seven image sequences listed in Table 3. Notice the use of impostors for group objects in the image sequences rendered with the hierarchical algorithm, and of null impostors in those rendered with the non-hierarchical algorithm.

Image Sequence Pairs

In the selection of image sequence pairs for comparison, we selected a subset of all possible pairs, in accordance with our intention of comparing the perception of image sequences rendered with both algorithms at the same rendering cost limit. A limited subset was chosen due to time constraints. Table 4 shows the image sequence pairs created from the image sequences in Table 3.

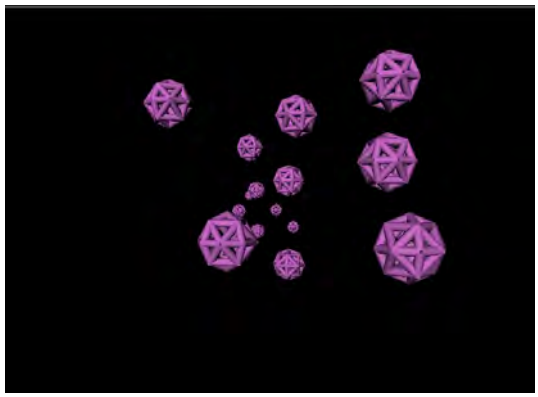


Figure 46: **First frame of image sequence 7.** The first frame of image sequence 7, rendered with the non-hierarchical algorithm and with a rendering cost limit equal to the sufficient cost of the scene, 15360. This represents the rendering of the entire scene at maximum level of detail.

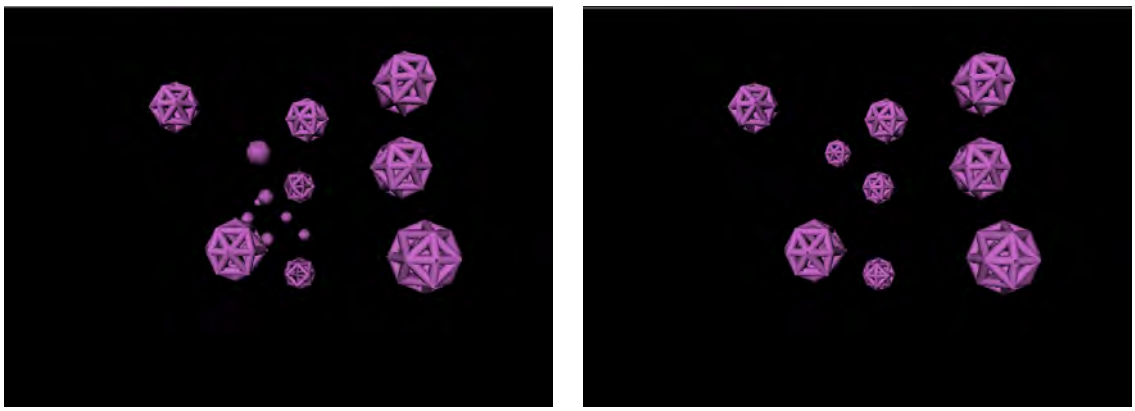


Figure 47: **First frames of image sequences 1 and 2.** The first frames of image sequences 1 (left) and 2 (right), rendered with the hierarchical and non-hierarchical optimization algorithms respectively. The rendering cost limit is 7680, twice the nominal cost of the scene. Notice the use of group and null impostors in the hierarchical and non-hierarchical cases respectively. Null impostors are visible as missing objects.

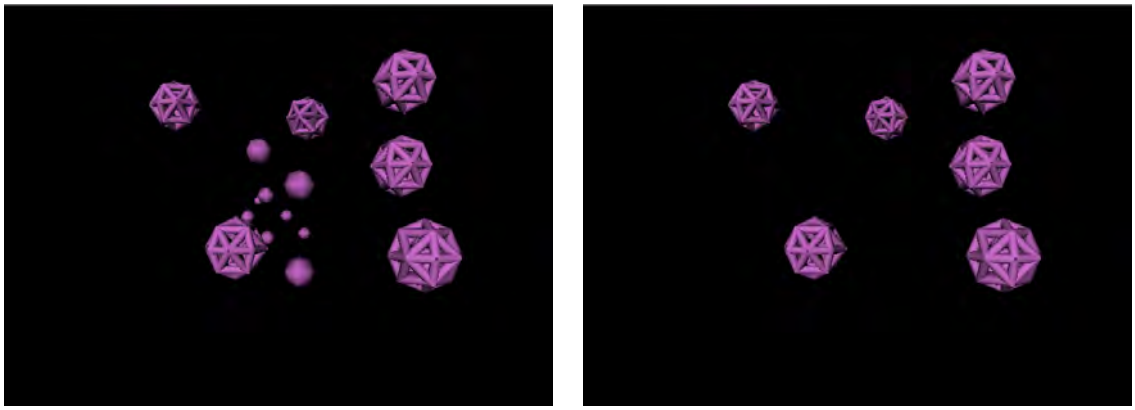


Figure 48: **First frames of image sequences 3 and 4.** The first frames of image sequences 3 (left) and 4 (right), rendered with the hierarchical and non-hierarchical optimization algorithms respectively, and with a rendering cost limit of 3840. This represents the behaviour of the two algorithms at a rendering cost limit equal to the nominal cost of the scene.

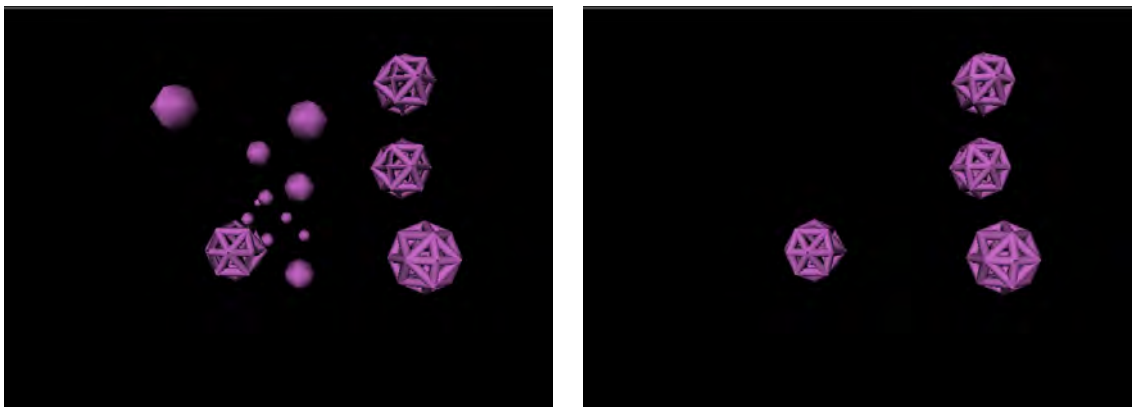


Figure 49: **First frames of image sequences 5 and 6.** The first frames of image sequences 5 (left) and 6 (right), rendered with the hierarchical and non-hierarchical algorithms respectively, and with a rendering cost limit of 1920, half the nominal cost. The extensive use of group and null impostors in the hierarchical and non-hierarchical cases respectively is apparent.

Sequence pair	First sequence	Second sequence
1	7	5
2	4	3
3	6	7
4	1	2

Table 4: **Image sequence pairs.** The pairs of image sequences used in the experiment.

Sequence pair 2 compares the perceptions of image sequences 3 and 4, which were rendered with different algorithms at a rendering cost limit of 3840. Sequence pair 4 compares the perceptions of image sequences 1 and 2, which were rendered with different algorithms at a rendering cost limit of 7680. Sequence pairs 1 and 3 compare the perceptions of image sequences 5 and 6 to the perception of image sequence 7, respectively. Taken together, they compare the perceptions of image sequences 5 and 6, which were rendered with different algorithms at a rendering cost limit of 1920. Recall that image sequence 7 was rendered at maximum level of detail for all objects, and is used here as a reference sequence.

8.2.4 Subjects

Fifteen assessors participated in the experiment, none of whom were knowledgable in the field of level of detail optimization, although some were knowledgable in computer graphics in general. They were screened for normal corrected visual acuity, and the results indicated that all had acceptable visual acuity. Some had limited previous experience as assessors, as the experiment was conducted in conjunction with that of another researcher, using an almost identical group of assessors.

8.2.5 Experimental Conditions

The image sequences were recorded frame by frame onto Hi8 video tape and presented on a single monitor in the form of a large television screen. The assessors were seated individually at a distance six times the height of the screen. The background illumination in the viewing room was provided by adjustable overhead lights, and was set to low.

8.2.6 Evaluation

A categorical judgment grading scale was used. With this form of grading scale, assessors are asked to assign the relation between each pair of stimuli to one of a set of semantically defined categories. The categorical scale used was that recommended by the CCIR [12], shown in Table 5.

Index	Description
-3	much worse
-2	worse
-1	slightly worse
0	the same
+1	slightly better
+2	better
+3	much better

Table 5: **Categorical grading scale.** The categorical grading scale used in the experiment.

The semantic description of each category refers to the relationship of the second sequence to the first. This grading scale yields a distribution of judgment indices across the scale categories for each pair of image sequences.

8.2.7 Procedure

In this section we describe the experimental procedure employed in this experiment.

Presentations

The image sequences constituting the stimulus material were grouped into pairs as described in Section 8.2.3. One session was held for each assessor, during which he or she viewed and voted twice on each pair. Each session lasted between twenty and thirty minutes, including all introductions of the assessors to the assessment method.

There were eight *assessment trials* per session, plus two test trials in which the assessors were able to practice the assessment method. Each assessment trial constituted the comparison of two image sequences. The structure of a trial is shown in Figure 50.

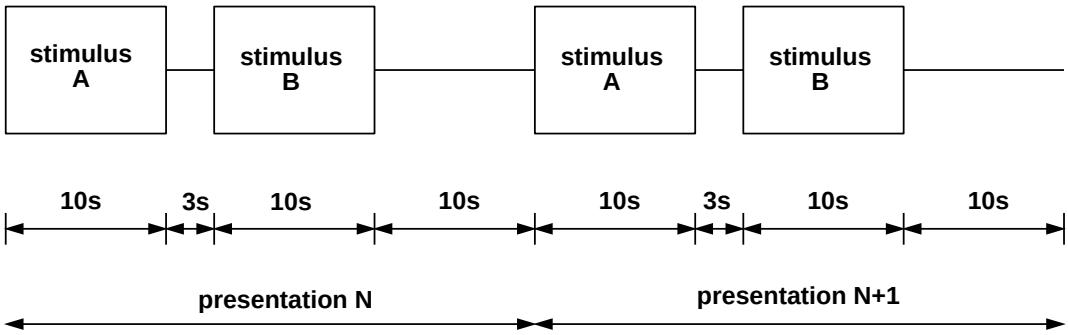


Figure 50: **Structure of an assessment trial.** Each trial consisted of two presentations. During each presentation stimulus A was shown for 10 seconds, then stimulus B, separated by a 3 second mid-grey adaption field and followed by a 10 second mid-grey post-adaption field.

Each trial consisted of two *presentations*. During each presentation stimulus fields A and B were each shown for 10 seconds. A mid-grey *adaption field* was shown for 3 seconds between the two stimulus fields and a mid-grey *post-adaption field* for 10 seconds afterwards. During the 10 second post-adaption field the number of the trial was superimposed in black. Voting on the trial was only allowed during the final post-adaption field of the trial. Only one display monitor was used.

The second four assessment trials were a repeat of the first four, showing the same image sequences again in the same order. However during these trials the assessors were allowed to view each trial as many times as they felt necessary to arrive at a final judgment. Therefore during a session an assessor provided two sets of indices of the same four assessment trials, with the first set being forced judgments after only one viewing, and the second being considered judgments after repeated viewing. The four unique assessment trials were ordered psuedo-randomly.

Introduction to Assessments

The assessors were introduced to the assessment methodology before the assessment trials of that session began. These introductions were provided in written form for consistency, although verbal questions were answered. Care was taken to avoid the introduction of bias.

In particular, the assessors were instructed as to the assessment procedure, the sequence and timing of presentations, the allowed voting period, and the grading scale used. The type and range of impairments likely to occur were described in the introduction and demonstrated by means of the

test assessment trials.

The assessors were told that all of the image sequences represented the same scene consisting of 16 objects, but drawn differently. They were explicitly asked to compare their perceptions of the two image sequences in each presentation, and to express this relation to the best of their ability in terms of the semantic grading categories provided.

8.2.8 Level of Detail Optimization Algorithms

Two level of detail optimization algorithms were compared: an early version of the incremental hierarchical optimization algorithm described in Chapter 7 and the non-hierarchical algorithm of Funkhouser and Séquin. The hierarchical algorithm is distinguished by its use of an hierarchical scene description and its support for impostors, or shared drawable representations, for group objects. Here we describe in detail the characteristics of the actual algorithm implementations used in the experiment.

Level of Detail Descriptions

Figure 51 shows conceptually the hierarchical level of detail description used in the case of the hierarchical optimization algorithm. The scene (root) object is a group object consisting of the group objects representing the domes, which in turn consist of the leaf objects representing the cylinders. The leaf objects and the dome group objects have associated explicit impostor representations, whereas the scene group object does not.

The level of detail description used in the case of the non-hierarchical algorithm is shown in Figure 52 and can best be thought of as a non-hierarchical collection of distinct and independent objects, each with its own associated impostors. These objects correspond exactly to the objects at the leaves of the hierarchy in the hierarchical description of the hierarchical optimization algorithm, and represent the individual cylinders making up the geodesic domes. There is no concept of group objects in the non-hierarchical algorithm, or of explicit shared levels of detail for them.

Levels of Detail

Six impostors were supplied for each leaf object in the hierarchical case and each object in the non-hierarchical case. Since all of these objects are cylinders, their impostors were created easily and naturally using the *SoComplexity* node provided by Inventor [52]. This node allows an Inventor

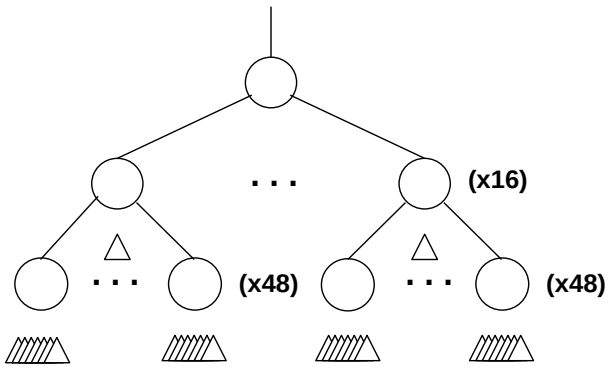


Figure 51: **Hierarchical level of detail description.** The level of detail description used in the case of the hierarchical optimization algorithm. The description consists of 768 objects representing cylinders, which are grouped into 16 groups of 48 cylinders each representing domes. The 16 dome group objects are then grouped into a single scene object. Each cylinder object is provided with 6 impostors. In addition the dome objects are each provided with a single impostor.

application to specify the geometric complexity, or subdivision level, of all *Inventor* primitives, including the *SoCylinder*. Figure 53 shows the six cylinder impostors used.

The levels of detail of the cylinder objects consist of the six cylinder impostors in the hierarchical case, and the six cylinder impostors plus a single null impostor in the non-hierarchical case. These levels of detail are shown in Table 6 and Table 7.

Null impostors were supplied for the cylinder objects in the non-hierarchical case, since we are interested in comparing the effects of the availability of impostors to the effects of not having impostors, and these effects are different mainly when the available rendering time is insufficient

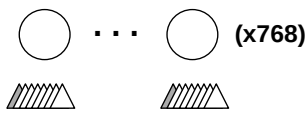


Figure 52: **Non-hierarchical level of detail description.** The level of detail description used in the case of the non-hierarchical optimization algorithm. The description consists of 768 distinct objects representing the cylinders of which the scene is composed. Each cylinder object is provided with 6 impostors plus a single "null" impostor (shaded). No shared impostors are provided for groups of objects.

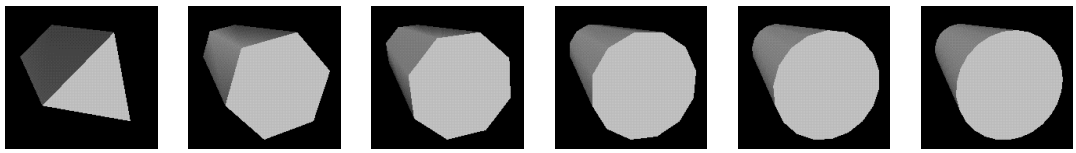


Figure 53: **Cylinder impostors.** The six cylinder impostors in order of increasing level of detail, corresponding to the *Inventor SoComplexity* values 0.00, 0.12, 0.20, 0.35, 0.53 and 0.61.

Level of detail	Description	Complexity value
LoD_1	lowest	0.00
LoD_2	.	0.12
LoD_3	.	0.20
LoD_4	.	0.35
LoD_5	.	0.53
LoD_6	highest	0.61

Table 6: **Cylinder object levels of detail for the hierarchical case.** Cylinder object levels of detail used with the hierarchical optimization algorithm, consisting of the six cylinder impostors. Note that the impostor corresponding to complexity value 0.00 is still a renderable geometric object, albeit a very simple one (See Figure 53).

to render all objects at even their lowest non-null levels of detail. In such situations, the non-hierarchical algorithm is only able to meet the rendering cost limit by selecting null impostors for some objects.

Recall from Section 8.1 that part of our hypothesis is that, because shared representations for groups of objects may be simpler than even the simplest separate representations for those objects, the use of shared representations allows the saving of additional rendering cost without resorting to the use of null impostors. For this reason we chose, for our lowest detail non-null impostors, representations that were about as simple as they could reasonably be made. As shown in Figure 53, the lowest detail impostors of the cylinder leaf objects consist of 3 sided prisms.

In the hierarchical case, a single impostor was provided for each group object corresponding to a geodesic dome in the scene. These constituted the single explicit levels of detail of the dome objects. The representation for the dome impostor was implemented as an SoSphere object in Inventor, with

Level of detail	Description	Complexity value
LoD_0	null	-
LoD_1	lowest	0.00
LoD_2	.	0.12
LoD_3	.	0.20
LoD_4	.	0.35
LoD_5	.	0.53
LoD_6	highest	0.61

Table 7: **Cylinder object levels of detail for the non-hierarchical case.** Cylinder object levels of detail used with the non-hierarchical algorithm, consisting of the six cylinder impostors and a single null impostor.

an SoComplexity value of 0.12. This representation was chosen because of the similarity of its appearance to that of the dome objects when represented implicitly by the explicit representations of their cylinder object children.

Heuristics

The benefit and cost heuristics provided for all objects were intentionally simplistic, and were intended to function as suitable test cases rather than as accurate predictions of the true contribution to scene perception and rendering cost of those objects.

The benefit heuristic, predicting the contribution to scene perception of a given object representation in both the hierarchical and non-hierarchical cases, was defined as follows:

$$\text{Benefit}(O, L) = \text{Accuracy}(O, L) * \text{ObjectSize}(O) * \text{ScreenSize}(O) \tag{48}$$

where:

- $\text{Accuracy}(O, L)$ is defined as predicting the "rendering accuracy" of the level of detail L of object O, and is given by:

$$\text{Accuracy}(O, L) = 1 - \frac{0.5}{L^{1.2}} \tag{49}$$

for a cylinder object at LoD_L, $L \neq 0$.

This model of representation accuracy was proposed by Funkhouser and Séquin [24], and provides increasing accuracy for more complex renderings, but to a diminishing extent.

The coefficient of 1.2 was selected based on a best fit against empirical estimates of the perceived accuracy of each representation. The constant 0.5 was chosen to satisfy the convexity assumptions of the algorithms (of descending relative value and descending value respectively — see Sections 2.6.1 and 7.3).

$$\text{Accuracy}(O, L) = 0.01 \tag{50}$$

for a cylinder object at null level of detail LoD_0.

The accuracy of the null cylinder representation was set to an arbitrary value of 0.01 which is just high enough to ensure that the convexity assumption is satisfied even for null levels of detail. In particular, we require that the value of any object at LoD_0 should be greater than the value of that object at LoD_1.

$$\text{Accuracy}(O, L) = 0.21 \tag{51}$$

for a dome object at explicit LoD_1, the dome impostor.

The accuracy of the non-leaf object impostor was similarly set to an arbitrary figure of 0.21, which is just high enough to ensure that the convexity assumption holds for all descendant objects that are represented by that impostor. In particular, we require that the value of a non-leaf object at any of its explicit levels of detail should be higher than the value of any of its descendant objects at their explicit levels of detail LoD_1.

- $\text{ObjectSize}(O)$ predicts the average projected area of an object O in object space, and is approximated by:

$$\text{ObjectSize} = \frac{\text{length} * \text{diameter}}{2} \tag{52}$$

for a cylinder object at any level of detail, and

$$\text{ObjectSize} = \pi * \text{radius}^2 \tag{53}$$

for a dome object at explicit LoD_1.

The size of an object, meaning the object space projected area of that object from the viewing direction, was approximated by a simple viewing direction independent scheme. Since all

leaf objects were cylinders, their average projected area in object space was approximated by half of the product of their length and diameter. The size of group objects, whose impostors were low-resolution spheres, was approximated by their cross-sectional area.

- ScreenSize(O) is inversely proportional to the square of the euclidean distance of object O from the camera, and is given by:

$$\text{ScreenSize} = 1/\text{distance}^2 \tag{54}$$

The ScreenSize of an object is an indication of the inverse of the degree to which apparent object area is diminished by physical distance. It is therefore approximated by the inverse of the square of the distance of the object from the camera. Euclidean distance was used for convenience of implementation in Inventor, although perpendicular distance from the viewplane would be more accurate [10].

The cost heuristic, predicting the rendering cost of a given object representation in both the hierarchical and non-hierarchical cases, was defined as follows:

$$\text{Cost}(O, L) = \text{Samples}(O, L) \tag{55}$$

where:

- Samples(O,L) is defined as approximating the number of "samples" (polygons, in this case) of the level of detail L of object O, and is given by:

$$\text{Samples}(O, L) = \begin{cases} 5 & \text{for } L = \text{LoD}_1 \\ 8 & \text{for } L = \text{LoD}_2 \\ 10 & \text{for } L = \text{LoD}_3 \\ 14 & \text{for } L = \text{LoD}_4 \\ 20 & \text{for } L = \text{LoD}_5 \\ 28 & \text{for } L = \text{LoD}_6 \end{cases} \tag{56}$$

for a cylinder object O at LoD_L, $L \neq 0$.

These values are approximations of the number of polygons used by Inventor for the corresponding impostor representations.

$$\text{Samples}(O, L) = 0.0001 \tag{57}$$

for a cylinder object O at null level of detail LoD_0.

The figure 0.0001 was chosen to provide a negligible but non-zero cost for each null LoD, so as to avoid undefined value calculations.

$$\text{Samples}(O, L) = 32 \tag{58}$$

for a dome object O at explicit level of detail LoD_1, the dome impostor.

This figure represents the number of polygons in the dome impostor, namely the Inventor SoSphere primitive at an SoComplexity value of 0.12.

This formulation of the cost heuristic is based loosely on that of Funkhouser and Séquin given in [24]. A more complete implementation of their cost heuristic might have taken into account other factors such as the dependency of the rendering cost of an object representation on the distance of that object from the viewer.

8.3 Results and Discussion

Figure 8 shows the results of the initial choice voting over the four assessment trials, while Figure 9 shows the results of the considered choice voting. Recall from Section 8.2.2 that trial T1 compared the full detail sequence with the hierarchical low cost (1920) sequence, trial T2 compared the non-hierarchical and hierarchical sequences for medium cost (3840), trial T3 compared the non-hierarchical low cost (1920) sequence with the full detail sequence, and T4 compared the hierarchical and non-hierarchical high cost (7680) sequences.

The results of the experiment appear to be significant. The 95% confidence intervals of all but the last trial exclude zero and therefore indicate a significantly positive or negative average value, in both the initial choice and considered choice sections.

The results in the initial choice and considered choice sections do not differ very significantly. In particular, they do not change sign. This suggests that the initial choice votes are reliable as indicators of the results of the experiment.

The second assessment trial compared the perceptions of the image sequences rendered with the non-hierarchical and hierarchical algorithms for a rendering cost limit equal to the nominal cost of

Trial	$\overline{x}$	sx	VAR[$\overline{x}$]	95% interval
T1	-1.067	1.387	0.128	[-1.783, -0.351]
T2	0.867	1.356	0.123	[0.167, 1.567]
T3	1.800	1.656	0.183	[0.945, 2.655]
T4	0.067	1.385	0.128	[-0.622, 0.757]

Table 8: **Voting indices for initial choice evaluation.** Shown is the average indicator value for each trial, as well as the standard deviation, variance and 95% confidence interval of the average value.

Trial	$\overline{x}$	sx	VAR[$\overline{x}$]	95% interval
T1	-1.133	1.642	0.180	[-1.988, -0.279]
T2	1.133	1.126	0.085	[0.552, 1.715]
T3	1.600	1.882	0.236	[0.628, 2.572]
T4	-0.067	1.438	0.138	[-0.809, 0.676]

Table 9: **Voting indices for considered choice evaluation.** Shown is the average indicator value for each trial, as well as the standard deviation, variance and 95% confidence interval of the average value.

the scene. The results of this trial suggest that, on average, the assessors considered the perception of the image sequence rendered with the hierarchical algorithm to be slightly better than that rendered with the non-hierarchical algorithm.

The fourth assessment trial compared the perceptions of the image sequences rendered with the hierarchical and non-hierarchical algorithms for rendering cost limits equal to twice the nominal cost of the scene. The results of this trial suggest that the assessors found no significant difference between the two image sequences, on average.

The first and third assessment trials compare the perceptions of image sequences rendered with the hierarchical and non-hierarchical algorithms respectively at a rendering cost limit equal to half the nominal cost of the scene to the image sequence rendered at maximum detail. Taken together, they compare image sequences 5 and 6. The results of these trials suggest that the assessors on average considered the impairment of perception due to the hierarchical algorithm to be less than that due to the non-hierarchical algorithm, at a rendering cost limit equal to half the nominal cost of the scene.

Taking these results together, we deduce that the assessors on average rated the perception of

the image sequences rendered with the hierarchical optimization algorithm to be better than that of those rendered with the non-hierarchical algorithm, in the cases where the rendering cost limit was equal to the nominal cost and half the nominal cost, but were unable on average to find a significant difference in the case where the limit was equal to twice the nominal cost. This suggests that the use of the hierarchical algorithm improves the perception of image sequences such as these when the available rendering time is very low, but has little effect when it is not. This is as we would expect, since the hierarchical algorithm differs in the use of group impostors instead of null impostors. These are only used at low rendering cost limits or, equivalently, when the visible scene complexity is high.

Furthermore, we can assume that no difference would be visible for higher rendering cost limits, since the hierarchical and non-hierarchical algorithms behave more similarly, rather than less similarly, for higher rendering cost limits (or equivalently, for lower visible scene complexity).

We speculate that the reason for the assessors' preference of the image sequences rendered with the hierarchical algorithm, for rendering cost limits less than and equal to the nominal cost, was the impression of objects appearing and disappearing caused by the use of null impostors in the image sequences rendered with the non-hierarchical algorithm. This creates the impression of flickering "holes" in the scene representation. Although there is some visible degradation of the image sequences rendered with the hierarchical algorithm due to the switching between dome and cylinder representations, this effect is, in our opinion, less disturbing than the complete disappearance of those objects that occurs in the non-hierarchical case.

It is unclear to what extent the increase in detail levels in important objects afforded by the selection of simple shared representations for unimportant group objects by the hierarchical algorithm influenced the results. For one thing, the complete omission of object representations in the non-hierarchical case affords even greater savings, at the expense of disturbing visual "hole" effects.

8.4 Conclusion

The experimental results show that the image sequences produced with the hierarchical predictive level of detail optimization algorithm were preferred by typical assessors, on average, to those rendered with the non-hierarchical algorithm, for rendering cost limits less than or equal to the nominal cost of the scene.

We conclude that the appropriate use of impostors for group objects, as allowed by hierarchical level of detail optimization, can lead to an improvement of the perception of image sequences over

those rendered with conventional Funkhouser and Séquin style non-hierarchical predictive level of detail optimization. The advantages of the use of impostors for group objects became visibly apparent to the assessors when the visible scene complexity was relatively high – at least high enough to cause the omission of object representations by the non-hierarchical algorithm of Funkhouser and Séquin. We believe this improvement in visual appearance is due to the reduction in the perception of flickering "holes", or of objects appearing and disappearing, caused by the use of null impostors in the non-hierarchical case. The experiment was not conclusive regarding the advantages gained by the application of saved rendering cost to improved renderings of more important objects.

More generally we conclude that perceptual evaluation, the subjective evaluation of image sequences by non-expert volunteer users, may be usefully employed to provide real-world data on the effectiveness of graphics algorithms.

The next chapter, Chapter 9, describes a second experiment. Whereas this experiment was designed to test the usefulness of hierarchical level of detail techniques in general, the next focuses on our hierarchical algorithm in particular. Also, whereas this experiment used subjective perceptual evaluation methods to test abstract effects such as user conviction, the next uses measurement and analysis of performance information to test the efficiency of our algorithm and provide validation of the theoretical time complexity analysis of Chapter 7.

Chapter 9

Radiosity Experiment

This chapter describes a very detailed second experiment which we proposed and supervised as a 4th year computer science honours project. The numerous measurements taken provide insight into the practical application of our method to real world situations. The extensive programming and testing as well as the recording of results and much of their analysis comprising the project was conducted by Shaun Nirenstein and Simon Winberg, two honours students under our supervision in the Department of Computer Science at the University of Cape Town. The test system made use of and incorporated an earlier radiosity simulation system designed and implemented by Adrian Secchia, an MSc student in the Department. Our role consisted of providing the underlying theory of the algorithm design and the supervision of the project as well as constant involvement with regard to issues arising in the implementation, optimization and evaluation of the system. The overall analysis of the results is also our own.

Whereas the first experiment (described in Chapter 8) was aimed at demonstrating the general usefulness of hierarchical level of detail optimization, this experiment is geared towards providing a convincing experimental demonstration of the applicability of the hierarchical level of detail optimization algorithm described in Chapter 7 to a real-world visualization and rendering problem.

We begin in Section 9.1 by describing the aims of the experiment. In Section 9.2 we describe the methodology employed, and in Section 9.3 we present and discuss the results. Finally we draw some conclusions in Section 9.4.

9.1 Aims

Our primary objective in this experiment is to test the hypothesis that the incremental hierarchical level of detail optimization algorithm described in Chapter 7 may be used to perform hierarchical level of detail optimization in realtime in a practical interactive visualization system. This hypothesis requires that the algorithm is successful in limiting frame preparation times to ensure interactive frame rates. Since the preparation of each frame involves both level of detail optimization and the rendering of the selected scene representation, the hypothesis requires that both operations may be performed in the available frame preparation time.

Funkhouser and Séquin [24] assume that rendering and level of detail optimization may be performed in parallel, with the result that the full available frame preparation time is available for both level of detail optimization and rendering. We expect that in practice this parallelization will be partial at best due to dependencies and delays. As Funkhouser and

For example, rendering of the selected scene representation may only begin once the final solution to level of detail optimization for that frame is known, since the selected representation of any object may in principle be incremented or decremented at any time during optimization. At best, level of detail optimization for a frame may be performed in parallel with the rendering of the selected representation for the previous frame. However if the interactivity of the system is measured as the delay between a user action and the reflection of the resulting changes on the display then from the point of view of this latency level of detail optimization and rendering must be assumed to take place in series.

Frame preparation time is dependent on both the time taken for level of detail optimization (which we call *optimization time*) and the time taken for the rendering of the scene representation selected by the level of detail algorithm (*rendering time*). Showing that the frame time is acceptable therefore implies showing not only that the rendering time is acceptable but also that the time taken by the optimization algorithm itself is acceptable and leaves sufficient time for rendering. While our theoretical analysis of the efficiency of our level of detail algorithm (Chapter 7) suggests that its worst-case time complexity is $O(n \log n)$, this alone says little about actual average and worst case execution times of the algorithm. Furthermore since the algorithm is incremental and takes advantage of frame-to-frame coherence it is important to investigate how its behaviour depends on the amount of coherence between successive frames, which the complexity analysis fails to address. Since the time complexity of the algorithm is not constant with respect to the size of its input it makes no sense to speak of absolute execution times without making reference to workloads. We

therefore aim to investigate how the optimization time as well as the rendering time depend in general on the complexity of the scene being rendered.

9.2 Methodology

In this section we describe the methodology employed in the experiment.

9.2.1 Level of Detail for Hierarchical Radiosity

Hierarchical radiosity is a physically-based rendering technique in which equations modeling the diffuse transfer of light between surfaces are solved numerically to produce shading intensity values for each surface. As an approximation the scene is modeled by flat polygons, or *patches*, and intensities are only calculated for the vertices of these patches. An initial scene description consisting of a relatively small number of flat *top-level polygons* is adaptively subdivided (as shown in Figure 54) according to estimates of perceptual importance to produce a final collection of patches that approximate the scene. The image quality of the resulting visualization therefore depends strongly on the local level of refinement of the patch hierarchy.

Secchia [74], among others, has proposed a perceptually-based refinement heuristic that predicts the visual importance of surfaces according to a simplistic model of human visual perception and exploits the exaggerated importance of edges such as shadow boundaries to visual perception. The illuminated patch hierarchies generated using this heuristic are characterized by higher levels of refinement in areas that are, in some sense, perceptually more important. We use the radiosity engine implemented by Secchia to generate input files for out system. Furthermore we make use of the perceptual information inherent in the adaptively refined radiosity hierarchy to exploit visual perception in the form of benefit heuristics that predict the visual importance of potential impostors, taking into account the presence of perceptually important edges as detected by Secchia's refinement heuristic.

Since radiosity rendering is performed as a pre-process to rendering the level of detail-like adaptive subdivision of the top-level polygons is view-independent. The perceptual refinement heuristic predicts the inherent perceptual importance of patches and can make no assumptions regarding the position or orientation of the viewer. Therefore each part of the scene must be subdivided to the maximum level of detail that might be required in any reasonable viewing situation. Our approach is novel in that instead of simply rendering the entire patch hierarchy at the highest level of refinement reached by the algorithm everywhere in the traditional fashion, we treat the patch hierarchy as a

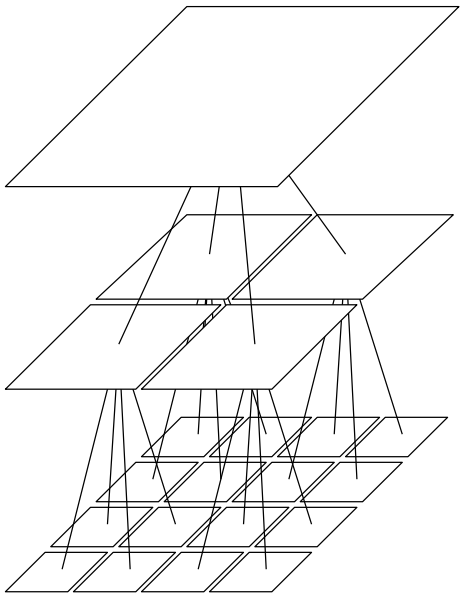


Figure 54: **Adaptive patch subdivision in hierarchical radiosity.** In hierarchical radiosity methods the polygons defining the initial coarse scene representation are recursively and adaptively subdivided into a *patch hierarchy*. Each patch is either retained or subdivided into smaller patches, depending on the outcome of a simple heuristic that predicts the complexity of the illumination function over that patch. In our system patches are quadrilaterals and are always subdivided into four equal-sized smaller quadrilaterals, if at all.

hierarchical level of detail description. The intermediate (non-leaf) patches that were generated and subsequently subdivided serve as low detail impostors for the patches that arose from them. This allows us to choose at render time the level of refinement appropriate for each part of the scene, taking into account the characteristics of the current viewing situation and the rendering time available.

Our hierarchical level of detail description consists of a hierarchy of nested patch objects. Each patch has a single polygon impostor, and its four children are the patches (if any) into which it was refined. The root object corresponds to the entire scene and has no impostor. Its children are the patches corresponding to the original top-level polygons. The level of detail optimization consists of the selection, for each frame, of a single subtree of the hierarchy rooted at the scene object. The polygon impostors at the leaves of the selected subtree comprise the selected scene representation. By taking advantage of view-dependent information about the position and orientation of the viewer

we are able to adaptively and dynamically favour increased patch resolution in areas that are perceptually more important. In addition, due to the predictive nature of the optimization algorithm we are able to place firm bounds on the predicted rendering cost of the selected scene representations. The aim is to render, for each frame, the most perceptually effective scene representation that may be rendered in the available rendering time. Note that the reduction of rendering complexity in unimportant areas allows us to render more important areas in increased detail. Figure 55 shows example output demonstrating the use of hierarchical level of detail optimization.

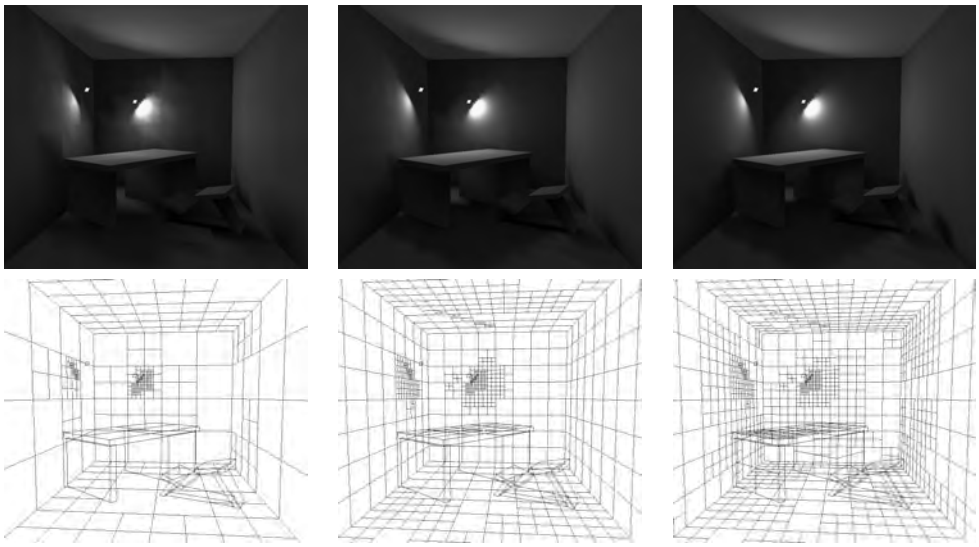


Figure 55: **Sample output of the experimental system.** The top three images show the same view of the same scene, with rendering cost limits equal to 500, 1000 and 1500 respectively. At bottom are wireframe renderings of the same views. Note the adaptive subdivision of polygons.

Note that the use of the non-leaf patches as polygon impostors without actively subdividing some neighbouring patches into triangles to resolve unshared vertices results in *T-vertex* artifacts (see Figure 56), visible as shading discontinuities. We chose for simplicity to ignore the T-vertex problem, with the result that some visual artifacts were introduced (see Figure 57). These could be avoided by triangular subdivision of offending patches if improved visual quality was desired.

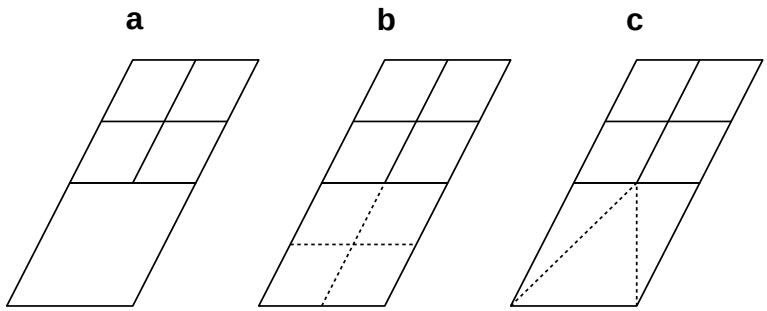


Figure 56: **The T-vertex problem.** The T-vertex problem arises when one of a pair of adjacent patches is subdivided for rendering and the other is not, as shown in (a). The most obvious solution, shown in (b), is to subdivide the adjacent patch as well. However this is impractical as it quickly propagates throughout the entire hierarchy, constraining the possible selections to those in which the entire hierarchy is subdivided to the same level. Instead it is possible to subdivide each adjacent patch into three triangular patches as shown in (c), which avoids the creation of new unshared vertices.

9.2.2 Scope

The size of the radiosity-generated scene descriptions used in the experiment is quite large: consisting of up to approximately 50000 polygons. The scenes represented are standard radiosity scenes often found in the literature (in particular, the `office` and `dining room` scenes [74]). The `office` and `dining room` scenes consist of around 200 and 400 top-level polygons respectively.

The choice of radiosity patches as objects in the hierarchical level of detail description, resulting in single-polygon impostors, represents a worst case for the level of detail algorithm since every single polygon must be individually considered for rendering by level of detail optimization. In the majority of visualization systems the objects in the hierarchy would be of a larger granularity and their impostors would typically consist of tens or hundreds of polygons, resulting in lower optimization overhead per polygon. By using the output of a hierarchical radiosity simulation on a polygon-per-impostor basis we essentially test the level of detail optimization algorithm in the extreme case in which every single scene primitive must be individually considered for rendering.

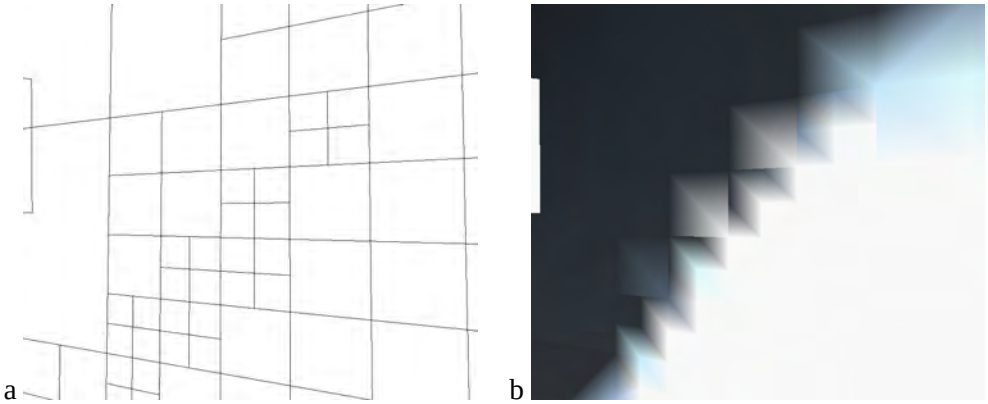


Figure 57: **An example of the T-vertex problem.** Images (a) and (b) show wireframe and shaded renderings of the same scene view. Note the occurance of T-vertex problems, identifiable in (a) as vertices that are incident to only 3 polygons and in (b) as shading discontinuities.

9.2.3 Experimental System

Nirenstein and Winberg [51] implemented an instrumented testbed system that allows the recording and playback of real and simulated user interaction during the realtime rendering of an optimized scene representation, as well as the measurement and recording of timing information. A screenshot of the system is shown in Figure 58.

In accordance with our aim of testing the time performance of our algorithm, the system allows the automatic recording of the optimization time, rendering time and total frame preparation time of each frame during a walkthrough of the scene using either an interactive or pre-recorded viewer path. It also allows the rendering of arbitrary scenes whose level of detail descriptions are generated using the hierarchical radiosity system of Secchia, adapted to also output the solution values of non-leaf patches.

The system is instrumented and allows the user to change the parameters of the optimization algorithm interactively, such as the *rendering cost limit* (the maximum permitted total cost of the selected representation for each frame) and the weightings of the components of the *benefit* and *cost* heuristics. It displays, in addition to the optimized scene representation rendered in OpenGL, an OpenGL wireframe rendering of the optimized representation and a graphical tree view of the hierarchical scene description showing which polygons were selected. The wireframe and hierarchy

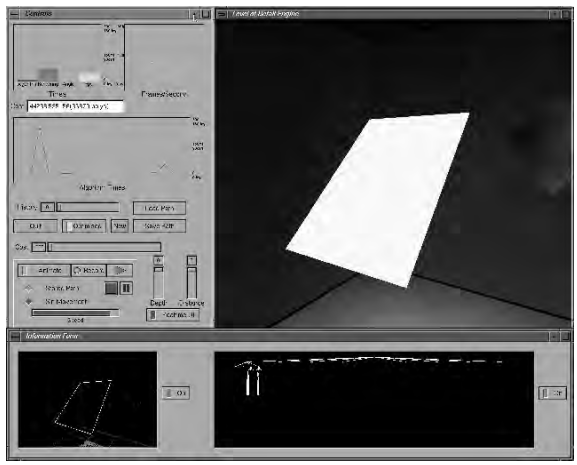


Figure 58: **Screenshot of the experimental system.** Screen shot of the test system displaying a radiosity scene. The top left window contains user interface controls and performance graphs. The top right window shows the rendered scene. The bottom window shows a wireframe view and the radiosity hierarchy.

views as well as the regular updating of the graphs and readings on the user-interface may be turned off to avoid compromising the accuracy of timing measurements.

All tests were performed on the same Silicon Graphics O2 workstation running the IRIX operating system. The workstation specifications are shown in Table 10.

Cost and *benefit* heuristics were provided that predict the rendering cost and perceptual benefit of object impostors. These heuristics were designed to be as simple as possible while still providing acceptable results. The rendering cost of our single 4 sided polygon impostors is measured as a constant 1.2 arbitrary units irrespective of viewing distance and size. We assume that since our polygons are typically relatively small their rasterization cost is relatively small and the rendering cost is therefore dependent mostly on their setup cost. Our results suggest that this is a sufficiently accurate approximation.

The benefit heuristic was formulated as:

$$\text{benefit}(p) = \text{depthConstant} * \text{depth} + \log(\text{sizeConstant} * \frac{\text{area}}{\text{distance}} + 1)$$

where *area* is the area in object space of the polygon comprising the impostor, *distance* is the distance of the center of the polygon from the viewer, and *depth* is the maximum depth of the full-detail hierarchical level of detail description at and below the node to which the impostor belongs. The *area* and *distance* measures provide an estimate of the projected size of the polygon on the

Component	Specification
System	SGI O2, IP32
Operating System	IRIX 6.3
Video subsystem	MVP version 1.4
Clock speed	175 MHz
FPU	MIPS R10010 Rev: 0.0
CPU	MIPS R10010 Rev: 2.5
Data cache	32 Kbytes
Instruction cache	32 Kbytes
Secondary cache	1 Mbyte on processor
Main memory	64 Mbytes

Table 10: **Test workstation specifications.** All tests were performed on a standard SGI O2 entry level workstation.

viewport (not taking into account changes in apparent size due to viewing direction, which were ignored). The effect of this is that patches close to the viewer are favoured over those that were further away. The constants determine the relative influence of the terms and can be interactively adjusted in our system. In addition we reduce the benefit of a polygon to zero if it is backfacing or if it is behind the viewer, in order to take advantage of the rendering cost saved by clipping and culling.

By biasing the benefit heuristic to predict greater perceptual benefit for polygons that were more densely refined in the maximum resolution scene description, we effectively encourage the selection of higher levels of detail in those areas. Since the patch hierarchy scene description was adaptively subdivided according to the perceptually-based refinement heuristic of Secchia [74] the maximum depth of the hierarchy at any point provides a convenient approximate measure of the perceptual importance (in terms of intensity gradients) of the detail at that point. The effects of the benefit heuristic are illustrated in Figure 59. Our simple viewplane-clipping approximation does not detect invisible detail which is in front of the viewer but outside the viewport, accounting for the fact that the amount of detail visible in Figure 59 (a) and (b) is greater than that in Figure 59 (c) and (d).

The fact that some areas of the scene are inherently more important than others (due, in this case, to the shading information they contain) became apparent after our first test runs with a simpler benefit heuristic based purely on distance. If important details such as shadow boundaries are treated the same as less important areas such as walls then the result is that the important details are rendered

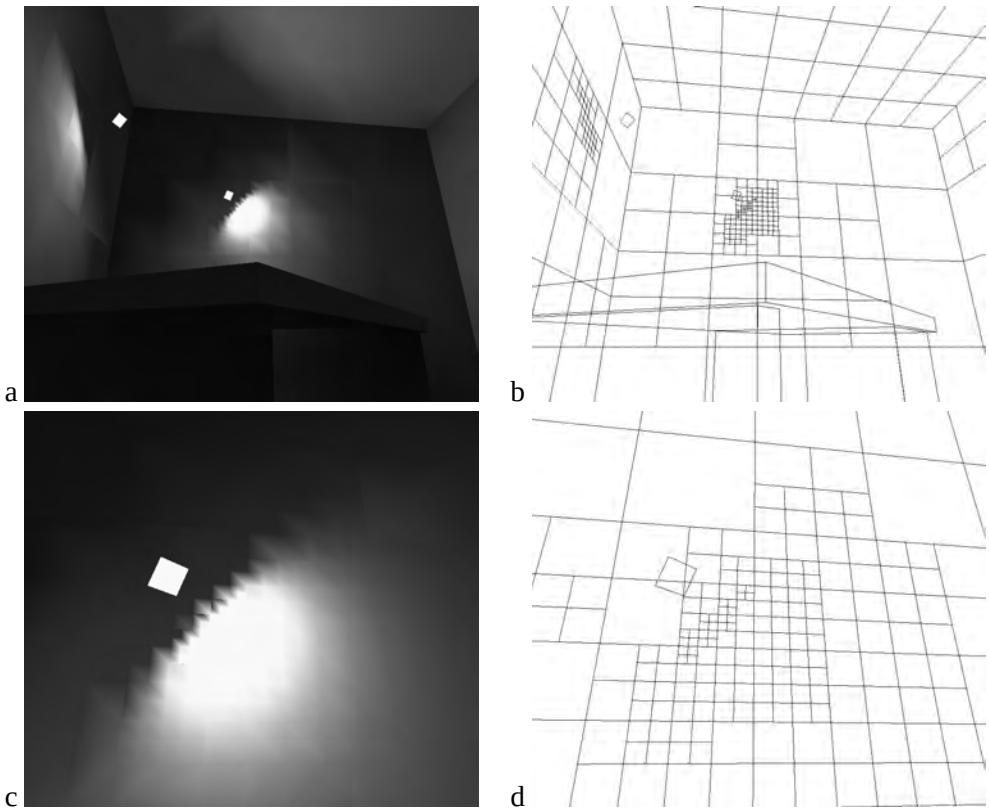


Figure 59: **Variation of local detail with viewing distance.** Images (a) and (b) show shaded and wireframe renderings of a scene view. Images (c) and (d) show shaded and wireframe renderings of the same view, zoomed such that the viewer is closer to the far wall. Note that the refinement of the polygons around the shadow boundary has increased as a result of the *distance* component of the benefit heuristic. Note also that the shadow boundary itself is well-refined in the far view. This is due to the *complexity* component of the heuristic, which favours higher detail in areas where the maximum detail is higher, ensuring that important areas such as shadow boundaries are given precedence over unimportant areas that are relatively uniform.

in poor detail in favour of better renderings of uninteresting expanses of wall. Some objects are inherently more interesting and require more detail to represent them adequately.

9.3 Results and Discussion

In this section we present the results of the experiment, together with discussion of their significance. We assume some familiarity with the workings of the hierarchical level of detail optimization algorithm, which was described in Chapter 7.

9.3.1 Dependence of Optimization Times on Changes in Viewing Angle

We began by investigating the consistency of the execution times of the optimization algorithm itself. Recall that the optimization algorithm is incremental and exploits frame-to-frame coherence by basing its initial solution on the solution found for the previous frame (Section 2). The success of this approach depends on the degree of coherence between the optimal solutions of consecutive frames. We therefore measured the change in viewing angle from one frame to the next (for simplicity, without regard to changes in viewing position) along each of several paths through the scene and noted the corresponding optimization times for each frame. Measurements were taken for four different *rendering cost limits*. The rendering cost limit dictates how much total detail the algorithm is allowed to select, and is measured in arbitrary cost units. Figure 60 shows the resulting graphs.

From Figure 60 it seems that the algorithm execution time is roughly proportional to the angular change in viewing direction between successive frames. This is to be expected as greater changes in viewing angle result in more objects becoming visible that were previously not visible and *vice versa*. As objects become newly invisible their allocated rendering cost must be reallocated amongst other objects (some of them newly visible) by means of repeated level of detail incrementations and decrementations.

Also evident is that the algorithm execution time is roughly proportional to the rendering cost limit. While the aim of the algorithm is to ensure constant *rendering* times irrespective of visible scene complexity, the optimization time (the execution time of the algorithm itself) increases as the amount of detail selected increases. Higher cost limits imply that more selected impostors must be considered for incrementation and decrementation in each iteration of the algorithm. This result is in contrast to the algorithm of Funkhouser and Séquin, whose execution time is independent of the rendering cost limit. Since our optimization algorithm is hierarchical it is able to save optimization

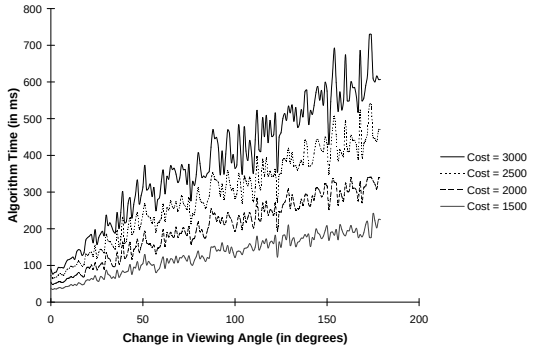


Figure 60: **Dependence of optimization times on turn magnitude.** Optimization algorithm execution times for various changes in viewing angle from one frame to the next along a typical path and for various rendering cost limits. The cost of a single impostor is 1.2 units. The "cost" referred to in the diagram is the rendering cost limit. See the text for an explanation of the high frequency variations that are evident.

time by making use of shared impostor representations that are more efficient to consider than a non-hierarchical collection of impostors providing the same number of levels of detail for each scene object. This saving increases as the rendering cost limit decreases, since lower detail impostors are shared to a greater extent than higher detail ones. The complexity of our algorithm approaches that of the Funkhouser-Séquin algorithm as the rendering cost limit tends to the cost of the full detail representation.

The high frequency irregularities of the graphs in Figure 60 can be explained by noting that the visibility of objects changes due to backface culling and clipping. Sometimes a small change in position or orientation will cause a top-level polygon refined into perhaps hundreds of patches to suddenly become visible where previously it was not. In these instances the allocation of rendering cost must be updated by means of repeated incrementations and decrementations. This destabilizing effect on optimization times is exaggerated in this particular case due to the fine granularity of the single-polygon impostors. In our scene the top-level polygons are generally each refined to hundreds or thousands of patches, so that large groups of impostors exhibit high visibility coherence.

The linear dependence of optimization time on change in viewing angle implies that the algorithm execution times are lower in cases with greater frame-to-frame coherence, as expected.

9.3.2 Frequency of Turn Magnitudes

Most of the turn magnitudes shown in Figure 60 represent pathological "non-incremental" cases in which the change in viewing angle is great and there is little coherence from one frame to the next. We therefore measured the relative frequency of turn magnitudes for a typical walkthrough in our system.

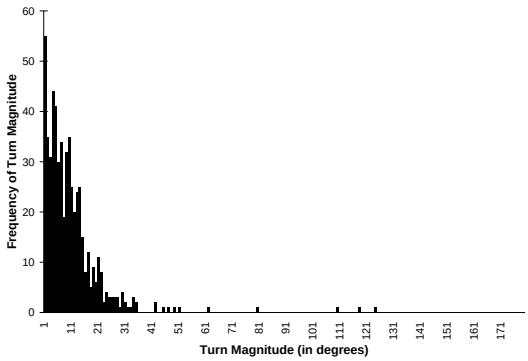


Figure 61: **Frequency of turn magnitudes.** The relative frequency of various changes in viewing angle between consecutive frames.

Figure 61 shows the resulting graph. It is clear that small changes in viewing angle greatly outnumber large ones, with changes above 30 degrees being extremely rare. We expect this to be the case in any useful interactive visualization system, as very large turn magnitudes are generally distracting to the user and in fact unlikely to occur at all at high frame rates. This argument suggests an important insight into the use of incremental techniques that exploit frame-to-frame coherence, namely that they may amplify any drops in frame rates that occur. This implies that it is more imperative than ever to ensure that reasonable frame rates are maintained. The greater the frame rate that is consistently maintained, the greater the interframe coherence and the better the performance of incremental rendering algorithms that depend upon it.

9.3.3 Algorithm Execution Times

The high degree of dependence of the algorithm execution time on frame-to-frame coherence and the relative infrequency of large turn magnitudes (and associated poor frame-to-frame coherence)

imply that the average execution time of the algorithm may be somewhat different to the worst case time. Indeed, this is the *raison d'être* of the incremental algorithm: to exploit frame-to-frame coherence and so ensure that average execution times are far better than worst case execution times, at the expense of the efficiency of the worst case. To test this we measured minimum, average and maximum optimization times for a typical path for a range of rendering cost limits.

Figure 62 shows the results. The average optimization time is closer to the minimum time than the maximum, and its behaviour is close to linear. We surmise that this is due to the relative infrequency of large turn magnitudes: typically there is significant coherence between successive frames. The minimum algorithm execution time (corresponding to the limit case in which consecutive frames are identical) is essentially constant with respect to the rendering cost limit, but the maximum (approaching the opposite limit in which consecutive frames are completely different) appears to be greater than $O(n)$.

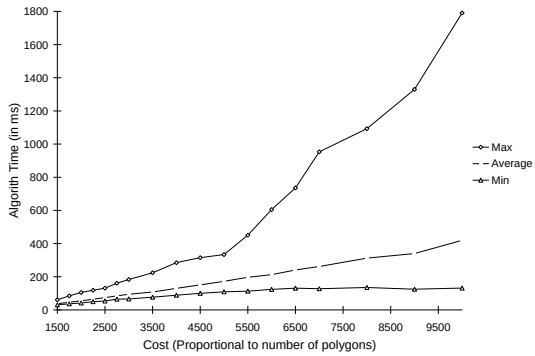


Figure 62: **Optimization algorithm execution times.** Plot showing how the maximum, minimum and average optimization algorithm execution times (over a typical walkthrough) vary with increasing rendering cost limit. The cost of a single polygon impostor is 1.2 units.

While our theoretical analysis (Section 7.3) predicts a theoretical worst case time complexity of $O(n \log n)$, it is unclear whether an $O(n \log n)$ implementation is *useful* in practice. The theoretical $O(n \log n)$ complexity assumes the use of ordered *priority queues* (as Funkhouser and Séquin [24] refer to them) of impostors available for incrementation and decrementation, reducing the complexity of the selection operations to $O(1)$ at the expense of update operations of $O(\log n)$. In the

implementation of our experimental system we found that the advantages of using advanced data structures with lower theoretical complexity (such as sorted trees rather than unordered lists) were questionable in this case due to the costs involved in maintaining them. Sorted trees for example would quickly become unbalanced and degenerate into lists if not actively rebalanced, since deletions always occur at the front. Instead we made use of unordered lists stored as static arrays, with a resulting algorithm complexity of $O(n^2)$. Static arrays provide an advantage that complex dynamic data structures cannot: they exploit cache coherence by ensuring that consecutively accessed data elements are always stored in consecutive areas of memory. The disadvantage of static arrays is that maintaining them in sorted order is expensive due to the constant need for shifting of elements to create space. For this reason we stored our lists as unsorted arrays. We found that the improvement of cache hit rate resulting from the use of arrays far outweighed the disadvantages of having to explicitly search the array for every selection.

The usefulness of the incremental algorithm hinges on the fact that consecutive frames generally exhibit a high degree of inter-frame coherence, as suggested by the relatively high frequency of small turn magnitudes shown in Figure 61. There are nonetheless cases in which coherence is limited and optimization times are significantly high. If left unchecked these may lead to excessive inter-frame delays due to the cost of the algorithm itself rather than the actual rendering. Considering Figure 62 again we see that for a maximum cost of 5500 (corresponding to approximately 4500 selected impostors, or optimization of more than 4500 scene objects for every frame) the maximum optimization time may be as high as 400 ms in pathological cases where inter-frame coherence is lacking.

9.3.4 Constancy of Frame Preparation Times

To test the constancy of frame preparation rates, we measured instantaneous frame rates (defined as the inverse of frame preparation time) for each frame of a typical walkthrough, with a rendering cost limit corresponding to 2500 selected impostors (or 2500 rendered polygons). In order to deduce the cause of any irregularities we found, we also measured the frame rendering times (excluding optimization times) and the optimization algorithm execution times for the same walkthrough.

Figure 63 shows the results. It is clear that frame preparation times vary dramatically from one frame to the next. We note however that the time taken to render the selected scene representation is relatively constant over all 160 frames, varying between approximately 50 and 80ms. This shows that the algorithm is successful in maintaining relatively constant rendering times. Furthermore, we note that optimization algorithm execution times vary dramatically from one frame to the next, and

that there is a marked correlation between the troughs in the graph of frame preparation times and the peaks in the graph of optimization times. This suggests that the variation in frame preparation time is dependent mainly on variations of the execution time of the optimization algorithm; the algorithm is successful in regulating frame rendering times, but is not guaranteed to take a limited or constant amount of time to do so. The objective thus becomes to place limits on the execution time of the optimization algorithm itself.

9.3.5 Truncation of Algorithm Execution

The inconsistency of frame optimization times, if left unchecked, might undermine the ability of the algorithm to regulate frame preparation times. We therefore implemented a simple cut-off scheme in which the algorithm's execution is simply halted if its execution time is found to have exceeded some predetermined limit. In the event of the algorithm being halted the solution reached so far is used as the final solution. Due to the iterative refinement strategy employed by the algorithm, its selected solution after any iteration always represents a feasible and complete (although not necessarily half-optimal) solution to the hierarchical level of detail optimization problem.

We measured the instantaneous frame rates achieved for a walkthrough with this technique. Figure 64 shows the results. It is clear that time-truncation of the optimization algorithm succeeds in ensuring a relatively constant frame rate, irrespective of visible scene complexity.

The disadvantage of truncating the algorithm execution time is that in the frames where the execution is truncated the algorithm produces a potentially less than half-optimal solution. This in theory results in occasional drops in visual quality. The amount of error introduced by truncation is approximately proportional to the amount of time truncated.

Since there is significant coherence not only between successive optimal detail levels but also between the changes in successive optimal detail levels over a series of frames, optimization time skipped on one frame is typically borrowed and will in the worst case need to be "repaid" in the form of additional computation in the following frames. The error introduced by truncation will always be corrected swiftly as long as excessive execution times are rare. In a typical system the image quality would worsen immediately after a sudden excessive motion by the viewer and then progressively improve (over at most a few frames) during periods of relatively little motion.

As we noted with regard to Figures 60 and 62, optimization time is dependent on the rendering cost limit and the degree of coherence between successive frames. Because the average optimization time is closer to the minimum optimization time than the maximum, we can expect the frequency of truncations to be relatively low.

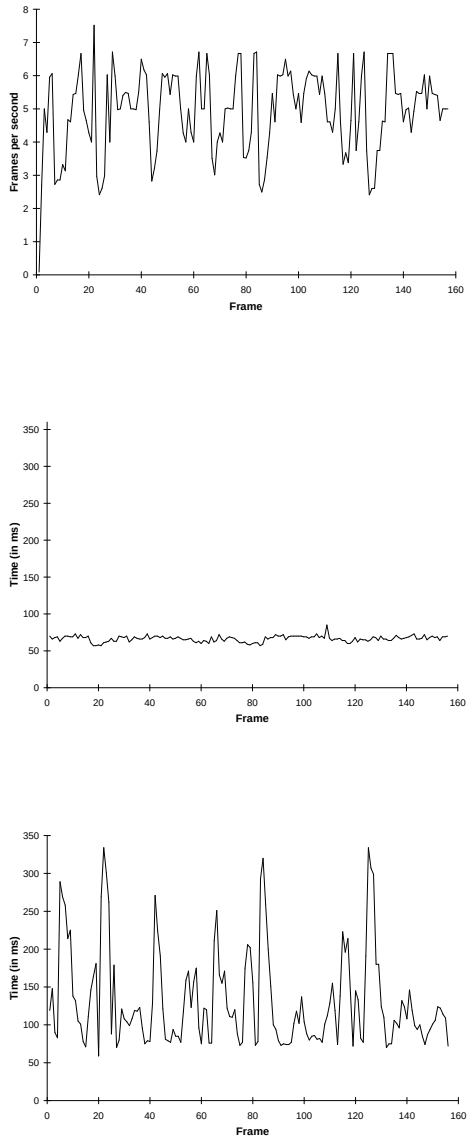


Figure 63: **Constancy of frame preparation times.** Plots showing (from top to bottom) instantaneous frame rates, frame rendering times (excluding optimization time) and optimization times (excluding rendering time) for each frame over the course of a typical walkthrough. The rendering cost limit is 3000, equating to 2500 single-polygon impostors.

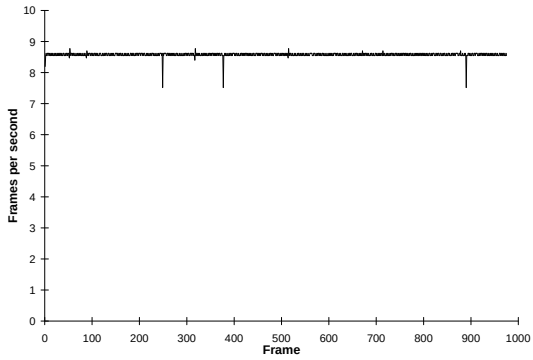


Figure 64: **Frame rates after truncation of optimization times.** Frame rates of a typical walkthrough (calculated as the inverse of individual frame preparation times), with optimization times truncated at 50ms. The cost limit is 1500, corresponding to 1250 single-polygon impostors. The full detail scene representation contains 36879 polygons.

9.3.6 Hierarchy Simplification

Recall from Figure 62 that the average optimization time was less than 100 ms for rendering cost limits lower than approximately 2500, corresponding to the selection of more than 2000 individual impostors. Because of the nature of our impostors, this corresponds to only around 2000 polygons. This fine granularity of one graphics primitive per impostor represents a worst case for our algorithm, since every single polygon in the scene must be individually considered for selection. In fact, due to the speed of the graphics hardware, the consideration of an impostor is more expensive than its rendering.

To improve this situation we implemented a *hierarchy simplification* strategy in the form of a transformation that reduces the hierarchy by collapsing multiple impostors into single shared representations. After application of this transform, the impostor of each object is the union of the impostors that previously belonged to its children. The leaves of the hierarchy are removed, as their impostors are now replaced by those of their parents. This transform is well suited to regular hierarchical descriptions such as this radiosity problem in which all the impostors are of the same type. In the instance of our hierarchy a single application of the transform results in each object (or patch) having a single impostor consisting of four polygons. A second application

results in impostors of sixteen polygons, and so forth. The general effect of the transform is to exponentially increase the granularity of the impostors so that more scene geometry is represented by each impostor. The cost and benefit heuristics must of course be adjusted accordingly.

To test the success of this approach we measured optimization times for a walkthrough of a scene after zero, one and two applications of the hierarchy simplification transform. The results are shown in Figure 65. The rendering cost limit in each case corresponds to a maximum selection of 1666 polygons. The result of applying the simplification transform is to greatly reduce the optimization time required to select the same amount of scene detail. After only one application of the transform the optimization times in Figure 65 are reduced to well below 25 ms for inter-frame turn magnitudes less than 50 degrees and for a selected scene representation consisting of around 416 impostors.

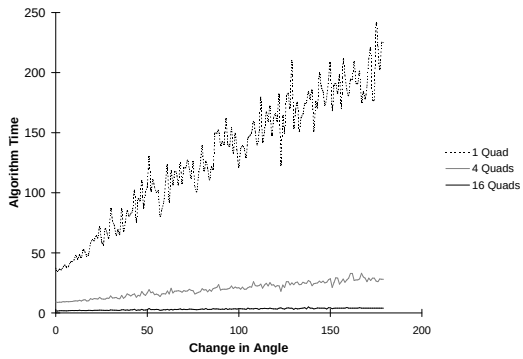


Figure 65: **Optimization times after hierarchy simplification.** A plot showing optimization algorithm execution times (averaged over four different walkthroughs of the same scene) for various changes in viewing angle after application of the hierarchy simplification transform zero, one and two times. The rendering cost limit in each case corresponds to 1666 selected polygons.

It is important to note that after the application of the transform (and adjustment of the cost heuristic to reflect the fact that impostors are now more expensive to render) the amount of detail that may be rendered within the available time does not change. Instead we have traded flexibility of detail selection for speed of optimization, by decreasing the number of possible combinations of impostors from which the algorithm may choose (see Figure 66). We have found in practice that a single application of the transform in our case results in an almost imperceptible loss of quality,

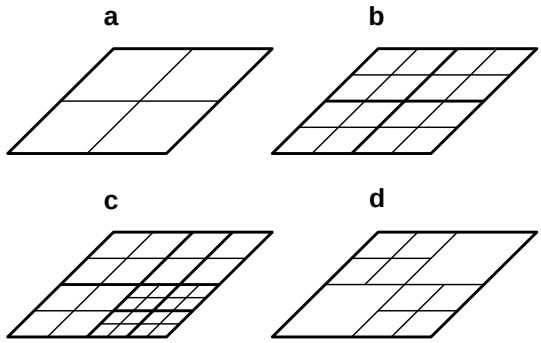


Figure 66: **Loss of flexibility due to hierarchy simplification.** Due to the aggregation of groups of impostors into combined, more complex impostors, the flexibility to select combinations of polygons for rendering is reduced. After one application of the simplification transform, all impostors contain four polygons. The single four-polygon impostor shown in (a) must be replaced by the four-polygon impostors of its four children as shown in (b). After that, any of those impostors may be replaced in turn by the impostors of its children, as shown in (c). However it is not possible to replace only part of the original four-polygon impostor, as shown in (d).

whereas two or more applications tend to result in visible degradation. Figure 67 compares the visible effects of zero, one and two applications of the transform.

9.3.7 Dependence of Frame Preparation Times on Scene Complexity

In order to test the dependence of frame preparation times (and therefore frame rates) on the complexity of the full detail scene, we measured non-optimized (full detail) rendering times, optimized rendering times, optimization times and optimized frame preparation times for identical walkthroughs of increasingly complex versions of the same scene, with the rendering cost limit held constant throughout.

Figure 68 shows the results. The unoptimized rendering renders the impostors at the leaves of the hierarchical scene description and the unoptimized rendering time increases linearly with the complexity of the scene description, as we would expect (since the number of leaf nodes in a regular hierarchy increases linearly with the total number of nodes). The rendering time of the optimized scene is constant irrespective of full detail complexity, as we would also expect since the complexity of the selected scene representation is dependent only on the constant rendering cost limit. The optimization algorithm execution times are constant except for low scene complexities

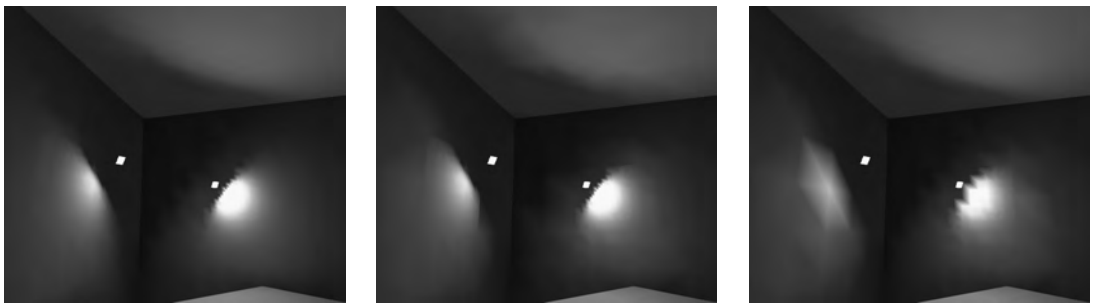


Figure 67: **Visual effects of the hierarchy simplification transform.** The same view of the same scene after zero, one and two applications of the hierarchy simplification transform.

where it increases with increasing scene complexity, probably due to more successful caching of smaller scene descriptions. The frame preparation time, being roughly the sum of the optimization time and the rendering time, behaves similarly to the optimization time and becomes constant for increasingly complex scene descriptions.

9.4 Conclusion

We presented the results of an experimental application of our predictive hierarchical level of detail optimization algorithm to the interactive rendering of hierarchical radiosity scenes. This work represents both an evaluation of the feasibility and effectiveness of the optimization algorithm and a demonstration of the applicability of hierarchical level of detail optimization to hierarchical radiosity.

The results attest to the predictive nature of the algorithm, showing that it may successfully be used to ensure bounded rendering times, to within the accuracy of the cost heuristic used. The algorithm selects a scene representation for every frame that may be rendered in the available time, regardless of the complexity of the full detail scene representation and the complexity of the visible portion of the scene.

We note that since the algorithm successfully regulates frame rendering times the variation in frame preparation times becomes dependent chiefly on the variable execution times of the algorithm itself. Moreover because the algorithm is incremental and successfully exploits frame-to-frame coherence its execution times are strongly dependent on the amount of coherence inherent in the

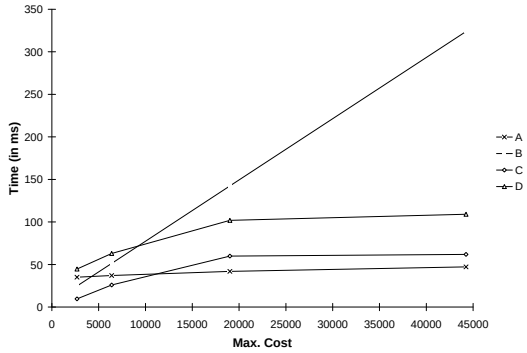


Figure 68: **Dependence of frame preparation times on scene complexity.** Graphs showing the effects of increasing full detail scene complexity (max cost) on (A) optimized rendering times, (B) unoptimized (full detail) rendering times, (C) optimization times and (D) optimized frame preparation times. The rendering cost limit is held constant throughout.

image sequence and are not guaranteed to be constant. In particular the frame-to-frame coherence on which the algorithm depends is strongly influenced by the effects of the angular change in viewing angle from one frame to the next on clipping and culling. The execution times of the optimization algorithm are far lower in the more common coherent cases than in the relatively rare cases in which coherence is lacking. In the worst case the complexity of the algorithm approaches $O(n \log n)$ or $O(n^2)$, depending on the implementation chosen, with respect to the number of scene impostors. In the best case it is $O(1)$. The average case is much closer to the best case than the worst and seems to be close to $O(n)$.

The application of the algorithm to the rendering of radiosity scenes holds promise. The most significant obstacles to the algorithm appear to be the fine granularity (in this case) of the level of detail description and the destabilizing effects of visibility culling on frame-to-frame coherence.

The strong dependence on frame-to-frame coherence exhibited by the algorithm suggests a useful lesson regarding the use of incremental algorithms in interactive systems: the use of algorithms that depend on frame-to-frame coherence for their efficiency serves to reinforce the need for consistent and reasonable frame rates. Any drops in frame rates that occur tend to cause the animation and user's input to be sampled at a lower rate, leading to a reduction in frame-to-frame coherence that

results in further performance degradation. The use of incremental algorithms such as ours makes it all the more important to ensure that reasonable frame rates are maintained.

Nirenstein and Winberg have demonstrated that a simple cut-off scheme may be used to prevent the algorithm execution times from exceeding an acceptable limit, while still providing a feasible and complete (though possibly occasionally non-half-optimal) solution for every frame. This ensures nearly constant frame rates and, as long as the cases in which the optimization time exceeds the limit are relatively rare, does not result in serious or cumulative degradation of the image sequence.

We note that the execution time of the algorithm is directly dependent on the number of impostors constituting its solution. This, in combination with the hierarchical nature of the algorithm, makes the optimization time dependent on the rendering cost limit, in contrast to the non-hierarchical algorithm of Funkhouser and Séquin which fails to make use of the increased efficiency of optimization resulting from shared impostors and so always performs the same amount of work for each incrementation and decrementation. In the case of this experiment the optimization time is linear with respect to the rendering cost limit, due to the identical cost of all object impostors.

The excessive optimization times recorded for the scene with single-polygon impostors suggest that, for the optimization algorithm to be useful in practice, the granularity of the impostors should not be too fine. A useful guideline is that the cost of considering an impostor for rendering should be significantly lower than the cost of simply rendering it without optimization. Nirenstein and Winberg have demonstrated the use of a *hierarchy simplification* transform that automatically increases the granularity of the impostors and so dramatically reduces the optimization effort required to optimize a given scene. After one application of this transform the algorithm was found to perform efficiently and with acceptable visual results.

Our experience suggests that rendering artifacts in important features such as edges and shadow boundaries are easily noticed and detract from user conviction even at relatively large distances. This suggests that some objects (or parts of objects) are inherently more important to perception, and disagrees with the popular assumption, implicit in static level of detail control based on distance, that all objects can be treated equally and that the perception of artifacts is predicted well by distance or screen-space size alone. It may often be worthwhile taking the inherent perceptual importance of objects into account in level of detail optimization.

The final result of our implementation was a working system in which the optimization algorithm was used to successfully regulate frame rates while providing acceptable levels of visual quality.

Chapter 10

Conclusion

In this thesis we have investigated the implications of hierarchical scene descriptions for predictive level of detail optimization. Our primary motivation is the development of improved predictive hierarchical level of detail optimization techniques that eliminate lag by actively regulating frame rendering times while optimizing the visual quality of the resulting frames. Our aims were to analyze the predictive hierarchical level of detail optimization problem formally, to develop tools for this formal analysis, and to derive and test improved predictive hierarchical level of detail optimization techniques.

We presented a new classification of existing level of detail optimization strategies, showing that the acceptance of predictive level of detail optimization has been surprisingly slow and that the application of predictive approaches to hierarchical scene descriptions has been all but nonexistent. In the light of this result, our aim is to investigating the reasons for this disparity and hence to address it. We reviewed previously demonstrated results showing that level of detail optimization is equivalent to a well-known constrained optimization problem, the *Multiple Choice Knapsack Problem* (MCKP). However we have shown that the equivalence is not as complete as was suggested, and in particular that it is compromised by the use of hierarchical scene descriptions with shared object representations. Furthermore we have drawn attention to errors in a previously proposed algorithm based on this approach that invalidate claims that it provides solutions with guaranteed perceptual quality levels and cast doubts on the solution quality of another algorithm based on it.

By basing our research on a more solid mathematical foundation and developing formal definitions of our ideas, we derived new algorithms and techniques that allowed us to correct the failings of previous approaches and overcome the difficulties posed by shared object representations. In

Chapter 4 we presented the first formal definition of a hierarchical level of detail description: a hierarchical scene description in which multiple shared representations may be provided for groups of objects. Using this formal definition as a basis we showed that the sharing of object representations that characterizes hierarchical descriptions effectively places implicit constraints on their selection. By making these constraints explicit we clearly identified the *hierarchical level of detail optimization problem*, and demonstrated its equivalence to a new hierarchical generalization of the MCKP, which we call the Hierarchical MCKP (Section 4.2). In the process we provided the first clear and formal distinction between the hierarchical and non-hierarchical level of detail optimization problems.

We developed a useful tool for the formal investigation of the hierarchical level of detail optimization problem. These *level of detail graphs*, described in Chapter 5, serve as visual and semantic representations of the state spaces generated by hierarchical level of detail descriptions. They are therefore useful in the analysis and investigation of the hierarchical level of detail optimization problem and associated optimization algorithms. In Section 7.2 we used them to prove the equivalence of the incremental and non-incremental versions of our predictive hierarchical level of detail optimization algorithm. In Section 7.2.4 we showed how this proof may be easily extended to serve as a proof of the equivalence of other previously presented algorithms whose equivalence was stated without proof.

Our main focus was the development of an improved predictive hierarchical level of detail optimization algorithm. This algorithm, presented in Chapter 7, is an extension and correction of previous approaches (namely those of Funkhouser and Séquin and Maciel and Shirley) that combines predictive level of detail optimization and the use of hierarchical scene descriptions, and therefore provides the benefits of both. It is predictive and so guarantees consistent and reasonable frame rendering times, making a significant contribution to the elimination of lag. It is hierarchical and so may take full advantage of the use of hierarchical scene descriptions with shared representations for groups of objects. It is incremental and exploits frame-to-frame coherence by basing its initial solution on the solution found for the previous frame. Our incremental algorithm is an extension of a new greedy approximation algorithm for the Hierarchical MCKP, which we presented in Chapter 6. We proved the correctness of the Hierarchical MCKP greedy algorithm, showing that its solution is always at least half-optimal for instances of a useful and well-defined subproblem of the Hierarchical MCKP. The level of detail optimization algorithm therefore provides guaranteed levels of predicted perceptual quality. In the process of developing the Hierarchical MCKP algorithm we presented and proved two new greedy algorithms for the conventional MCKP, whose correctness

we proved formally using mathematical techniques. One of these is a simplified version of the other that we proved is half-optimal under a simplifying assumption.

We presented the results of experiments testing the usefulness of hierarchical level of detail optimization in general and the effectiveness and efficiency of the hierarchical level of detail optimization algorithm in particular. Our first experiment, described in Chapter 8, introduced the use of *perceptual evaluation*, the subjective evaluation of image sequences by non-specialist users, for the evaluation of computer graphics algorithms. The second experiment, described in Chapter 9, consisted of the implementation of our hierarchical level of detail algorithm in a practical interactive visualization system and demonstrated its use in the realtime optimization of thousands of scene objects. One important contribution of this experiment was the first application of hierarchical level of detail optimization to the dynamic view-dependent adaptive simplification of radiosity-generated scene descriptions at render-time. The experiment demonstrated the effectiveness of the predictive hierarchical approach and the feasibility of our algorithm.

In this thesis we have presented an effective approximation algorithm for the predictive hierarchical level of detail optimization problem. Therefore an effective and feasible algorithm exists for the automatic selection of hierarchically defined detail levels with the aim of optimizing predicted visual quality while limiting predicted rendering cost. Our algorithm depends on the provision of reasonably accurate benefit and cost heuristics that predict the perceptual benefit and rendering cost of object representations. This effectively shifts the unsolved portion of the broader level of detail problem to the creation of accurate and efficient prediction heuristics. Since the algorithm successfully limits predicted rendering cost and provides guaranteed levels of predicted visual quality, the problem is to accurately predict the perceptual benefit and rendering cost of arbitrary object representations. Experience suggests that simple *ad hoc* heuristics tend to produce reasonable results. However this is still an open problem with much scope for further improvement.

Bibliography

[1] D. G. Aliaga. Visualization of complex models using dynamic texture-based simplification. In *IEEE Visualization '96*, pages 101–106. IEEE, October/November 1996.

[2] D. G. Aliaga and A. A. Lastra. Architectural walkthroughs using portal textures. In *IEEE Visualization '97*, pages 355–362. IEEE, October 1997.

[3] D. G. Aliaga and A. A. Lastra. Smooth transitions in texture-based simplification. *Computer and Graphics, Elsevier Science*, 22(1):71–81, 1998.

[4] J. Amanatides. Realism in computer graphics: a survey. *IEEE Computer Graphics and Applications*, 7(1):44–56, 1987.

[5] R. D. Armstrong, D. S. Kung, P. Sinha, and A. A. Zoltners. A computational study of a multiple-choice knapsack algorithm. *ACM Transactions on Mathematical Software*, 9:184–198, 1983.

[6] M. Beigbeder and G. Jahami. Managing levels of detail with textured polygons. In *Compugraphics '91, First International Conference on Computational Graphics and Visualization Techniques*, volume I, pages 479–489, 1991.

[7] S. Belblidia and J.-C. Paul. Generating various levels of detail of architectural objects for image-quality and frame-rate control rendering. *Computer Graphics International*, 1996.

[8] S. Belblidia, J.-P. Perrin, and J.-C. Paul. Multi-resolution rendering of architectural models. In *International Conference on Computer-Aided Architectural Design Features*, 1995.

[9] E. H. Blake. A metric for computing adaptive detail in animated scenes using object-oriented programming. In *Computer Graphics Forum (Eurographics)*, 1987.

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

[11] D. A. Carlson and J. K. Hodgins. Simulation levels of detail for real-time animation. In W. Davis, M. Mantei, and V. Klassen, editors, *Graphics Interface*, pages 1–8, May 1997.

[12] 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.

[13] A. Certain, J. Popović, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *Computer Graphics Proceedings, Annual Conference Series*, pages 91–98. ACM SIGGRAPH, 1996.

[14] 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.

[15] A. K. Chandra, D. S. Hirschberg, and C. K. Wong. Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science*, 3:293–304, 1976.

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

[17] L. DeFloriani, B. Falcidieno, C. Pienovi, D. Allen, and G. Nagy. A visibility-based model for terrain features. In *Proceedings of International Symposium on Spatial Data Handling*, July 1986.

[18] K. Dudzinski and S. Walukiewicz. A fast algorithm for the linear multiple-choice knapsack problem. *Operations Research Letters*, 3(4):205–209, October 1984.

[19] C. Erikson and D. Manocha. Simplification culling of static and dynamic scene graphs. Technical Report TR98-009, Department of Computer Science, University of North Carolina at Chapel Hill, 1998.

[20] J. S. Falby, M. J. Zyda, D. R. Pratt, and R. L. Mackey. NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation. *Computers and Graphics*, 17(1):65–69, 1993.

[21] M. L. Fisher. Worst-case analysis of heuristic algorithms. *Management Science*, 26(1):1–17, January 1980.

[22] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice, 2nd ed.* Addison-Wesley, Reading MA, 1990.

[23] H. Fuchs, D. P. Greenberg, and A. van Dam. Picture this: The changing world of graphics. In *Defining a Decade: Proceedings of CSTB's 10th Anniversary Symposium*. National Academy Press, 1996.

[24] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Computer Graphics Proceedings Annual Conference Series*, volume 27, pages 247–254. ACM SIGGRAPH, August 1993.

[25] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[26] G. Gens and E. Levner. An approximate binary search algorithm for the multiple-choice knapsack problem. *Information Processing Letters*, 67(5):261–265, 1998.

[27] A. S. Glassner. Adaptive precision in texture mapping. In *Computer Graphics*, volume 20, pages 297–306. ACM SIGGRAPH, 1986.

[28] E. B. Goldstein. *Sensation and Perception*. Brooks/Cole Publishing Company, 5 edition, August 1998. ISBN 0534346804.

[29] R. Gossweiler. A system for application-independent time-critical rendering. http://www.cs.virginia.edu/ rg3h/sigChiPaper.html.

[30] N. Greene, M. Kass, and G. Miller. Hierarchical z-buffer visibility. In *Computer Graphics*, volume 27, pages 231–238. ACM SIGGRAPH, 1993.

[31] R. Hamberg and H. de Ridder. Continuous assessment of perceptual image quality. *Journal of the Optical Society of America*, 12(12):2573–2577, December 1995.

[32] P. S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, 1986.

[33] P. S. Heckbert. *Simulating Global Illumination Using Adaptive Meshing.* PhD thesis, University of California, Berkeley, 1991.

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

[35] H. Hoppe. Progressive meshes. In *Computer Graphics Proceedings of SIGGRAPH '96*, volume 30, pages 99–108. ACM SIGGRAPH, 1996.

[36] H. Hoppe. View-dependent refinement of progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series*, pages 189–198. ACM SIGGRAPH, 1997.

[37] E. Horvitz and J. Lengyel. Perception, attention, and resources: A decision-theoretic approach to graphics rendering. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, August 1997.

[38] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, September 1995.

[39] T. Ibaraki, T. Hasegawa, K. Teranaka, and J. Iwase. The multiple-choice knapsack problem. *Journal of the Operations Research Society of Japan*, 21(1), March 1978. 59–95.

[40] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. Assoc. Comput. Mach.*, 22:463–468, 1975.

[41] C. Jackins and S. L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14(3):249–270, 1980.

[42] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, November 1979.

[43] P. Lindstrom, D. Koller, L. F. Hodges, W. Ribarsky, N. Faust, and G. Turner. Level-of-detail management for real-time rendering of phototextured terrain. Technical Report GIT-GVU-95-06, College of Computing, Georgia Institute of Technology, 801 Atlanta Drive, NW, 1993.

[44] P. Lindstrom, D. Koller, L. F. Hodges, W. Ribarsky, N. Faust, and G. Turner. Real-time, continuous level of detail rendering of height fields. In *Computer Graphics Proceedings, Annual Conference Series*, pages 109–118. ACM SIGGRAPH, 1996.

[45] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. Technical Report TR98-029, Department of Computer Science, University of North Carolina at Chapel Hill, August 1998.

[46] P. W. C. Maciel. Interactive rendering of complex 3D models in pipelined graphics architectures. Technical report, Department of Computer Science, Indiana University, Bloomington, 1994.

[47] 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.

[48] D. Marshall, D. S. Fussell, and I. A.T. Campbell. Multiresolution rendering of complex botanical scenes. In *Graphics Interface '97*, pages 97–104, May 1997.

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

[50] D. Meagher. Octree: A new technique for the representation, manipulation and display of arbitrary 3D objects by computer. Technical Report IPL-TR-80-111, Rensselaer Poltechnic Institute, 1980.

[51] S. Nirenstein and S. Winberg. Hierarchical level of detail optimisation algorithm evaluation. Technical Report CS98-15-00, Department of Computer Science, University of Cape Town, 1998.

[52] Open Inventor Architecture Group. *Open Inventor C++ Reference Manual*. Addison-Wesley Publishing Company, 1994.

[53] J. Pajon, Y. Collenot, X. Lhomme, N. Tsingos, F. Sillion, P. Guilleteau, P. Vuyslteker, G. Grillon, and D. David. Building and exploiting levels of detail: An overview and some VRML experiments. In *VRML '95 First Annual Symposium on the Virtual Reality Modelling Language*, pages 117–122, 1995.

[54] E. Puppo and R. Scopigno. Simplification, LOD and multiresolution: Principles and applications. Eurographics '97 Tutorial T4, Budapest, Hungary, September 1997.

[55] M. M. Rafferty, D. G. Aliaga, V. Popescu, and A. A. Lastra. Images for accelerating architectural walkthroughs. *IEEE Computer Graphics and Applications*, November/December 1998.

[56] R. L. Read, D. S. Fussel, and A. Silberschatz. System-wide multiresolution. Technical Report TR-93-04, Department of Computer Science, University of Texas at Austin, Austin, Texas 78712-1188, February 1993.

[57] M. Reddy. Reducing lags in virtual reality systems using motion-sensitive level of detail. In *Proceedings of the 2nd UK VR-SIG conference*, 1994.

[58] M. Reddy. Musings on volumetric level of detail for virtual environments. *Virtual Reality: Research, Development and Application*, 1(1):49–56, 1995.

[59] M. Reddy. A perceptual framework for optimising visual detail in virtual environments. In *Proceedings of the FIVE'95 Conference, QMW, University of London, 18-19 December*, 1995.

[60] M. Reddy. A survey of level of detail support in current virtual reality solutions. *Virtual Reality: Research, Development and Application*, 1(2):85–88, 1995.

[61] M. Reddy. A measure for perceived detail in computer-generated images. Technical report, Department of Computer Science, University of Edinburgh, 1996.

[62] M. Reddy. The development and evaluation of a model of visual acuity for computer generated imagery. Technical Report ECS-CSG-30-97, Department of Computer Science, University of Edinburgh, 1997.

[63] M. Reddy. The effects of low frame rate on a measure for user performance in virtual environments. Technical report, Department of Computer Science, University of Edinburgh, 1997.

[64] W. T. Reeves. Physically based modeling vs. faking it. *Communications of the ACM*, 31(2):116–117, 1988.

[65] J. Rohlf and J. Helman. Iris performer: A high performance multiprocessing toolkit for real-time 3D graphics. In *Computer Graphics Proceedings, Annual Conference Series*, volume 28, pages 381–394. ACM SIGGRAPH, July 1994.

[66] J. A. J. Roufs and H. Bouma. Towards linking perception research and image quality. In *Proceedings of the SID*, volume 21, pages 247–270, 1980.

[67] S. M. Rubin. The representation and display of scenes with a wide range of detail. *Computer Graphics and Image Processing*, 19:291–298, 1982.

[68] S. M. Rubin and T. Whitted. A 3-dimensional representation for fast rendering of complex scenes. In *Computer Graphics*, volume 14, pages 110–116. ACM SIGGRAPH, 1980.

[69] S. Sahni. Approximate algorithms for the 0-1 knapsack problem. *Journal of ACM*, 22:115–124, 1975.

[70] G. Schaufler. Exploiting frame to frame coherence in a virtual reality system. In *VRAIS '96*, pages 95–102, Santa Clara, California, April 1996.

[71] G. Schaufler. Image-based object representation by layered impostors. In *Symposium on Virtual Reality Software and Technology '98*, pages 99–104, November 1998.

[72] G. Schaufler, T. Mazuryk, and D. Schmalstieg. High fidelity for immersive displays. Technical Report TR-186-2-96-02, Institute for Computer Graphics, Technical University of Vienna, 1996.

[73] G. Schaufler and W. Sturzlinger. A three dimensional image cache for virtual reality. In *Computer Graphics Forum*, volume 15, pages 227–236. EUROGRAPHICS, August 1996.

[74] A. Secchia. Perceptual refinement for hierarchical radiosity. Master's thesis, Department of Computer Science, University of Cape Town, April 1998.

[75] 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.

[76] Silicon Graphics, Inc. *IRIS Performer Programmers Guide (Document Number 007-1680-030)*, 1995.

[77] Silicon Graphics, Inc. OpenGL optimizer. White paper, 1997.

[78] F. Sillion, G. Drettakis, and B. Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. In D. Fellner and L. Szirmay-Kalos, editors, *Computer Graphics Forum, proceedings of Eurographics '97*, volume 16, pages 207–218. Eurographics, Blackwell Publishers, 1997.

[79] G. J. F. Smets and K. J. Overbeeke. Trade-off between resolution and interactivity in spatial task performance. *IEEE Computer Graphics and Applications*, pages 46–51, September 1995.

[80] S. J. Teller and C. H. Sequin. Visibility preprocessing for interactive walkthroughs. In *Computer Graphics*, volume 25, pages 61–69. ACM SIGGRAPH, July 1991.

[81] K. Teunissen and H. D. M. Westerink. A multidimensional evaluation of the perceptual quality of television sets. *SMPTE Journal*, pages 31–38, January 1996.

[82] J. Torborg and J. T. Kajiya. Talisman: Commodity realtime 3D graphics for the PC. In *Computer Graphics*, volume 30. ACM SIGGRAPH, 1996.

[83] The VRML Consortium Incorporated. *VRML97, International Standard ISO/IEC 14772-1:1997*, 1997.

[84] B. Watson, N. Walker, and L. F. Hodges. A user study evaluating level of detail degradation in the periphery of head-mounted displays. Technical Report 95-31, Graphics, Visualization and Usability Center, Georgia Institute of Technology, 1995.

[85] B. Watson, N. Walker, L. F. Hodges, and A. Worden. Effectiveness of peripheral level of detail degradation when used with head mounted displays. Technical Report 96-04, Graphics, Visualization and Usability Center, Georgia Institute of Technology, 1996.

[86] C. Wiley, I. A.T. Campbell, S. Szygenda, D. Fussell, and F. Hudson. Multiresolution BSP trees applied to terrain, transparency, and general objects. In *Graphics Interface '97*, pages 88–96, May 1997.

[87] M. Wloka. Incorporating update rates into today's graphics systems. Technical Report CS-93-56, Department of Computer Science, Brown University, Providence, RI, USA, December 1993.

[88] H. Zhang. *Effective Occlusion Culling for the Interactive Display of Arbitrary Models*. PhD thesis, Department of Computer Science, UNC-Chapel Hill, July 1998.

[89] H. Zhang, D. Manocha, T. Hudson, and K. E. Hoff. Visibility culling using hierarchical occlusion maps. In *SIGGRAPH '97*, volume 31, 1997.