Adaptive Levels of Detail for Interactive Collision Handling

A Thesis

Submitted to the Office of Graduate Studies

of

Trinity College Dublin

in Candidacy for the Degree of

Doctor of Philosophy

November, 2002

by John Lalzoliana Dingliana

Declaration

[illegible]

Signature of Author

Acknowledgements

I am infinitely grateful to Carol O'Sullivan, my supervisor, without whom I would have fallen by the wayside a long time ago; for her belief and advice and countless other contributions to this thesis and to my career. I would like to thank all my colleagues, past and present, for sharing with me knowledge, inspiration and many hours of their lives. In particular, I must mention (by virtue of working in related areas) Thanh and Gareth for their much superior depth-first implementations of things that I only hoped possible, and Carol again for her constant advice on everything in general and perceptual issues in particular; not to mention proof-reading with time-critical effectiveness everything that I have ever written during the course of my research. I have to thank Professor John Byrne, who let me into the department and made life easy for me along the way. I also thank all my friends and family, who know just about enough of what I do to forgive me for neglecting them through the last stages of my work on this thesis. Last but not least, I thank my parents for their financial and spiritual support, without which I would never even have made it to the start line.

Abstract

Collision Handling has long been a major bottleneck in physically based animation. As scene complexity increases the problem becomes critical enough to prohibit real-time performance in the animation system. Previous approaches have attempted to achieve interactive rates by culling large parts of the scene or making drastic simplifications to the scene.

We present here an adaptive level of detail approach to the collision-handling problem for interactive applications, which guarantees real-time rates irrespective of the complexity of the scene provided that the underlying hardware is able to cope with the lowest level of detail. This is achieved by using interruptible methods for the three problems of Collision Detection, Contact modelling and Collision Response. A multi-resolution bounding volume hierarchy is used to provide discrete levels of resolution in the data structures used for collision detection. We present a perceptually based collision prioritisation approach which makes graceful degradation of collision detail possible across a complex virtual environment. We present strategies for facilitating trade-offs between accuracy and speed in Collision Handling in the attempt to enhance the immersive nature of the virtual environment and optimise the quality of the dynamic scene.

We investigate the use of different classes of bounding volume hierarchies in an interruptible collision handling scheme as well as various strategies for scheduling the traversal of the bounding volume trees in order to better optimise the speed-accuracy trade-off. We also present an approach for stochastic contact modelling and collision response based on density values of bounding volume nodes. An evaluation of the effectiveness of different approaches in terms of efficiency and plausibility is presented.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The science of computer graphics has always captured the imagination of researchers and, as a result, has evolved at a pace that has pushed the limits of available hardware. As new capabilities are so quickly exploited in software solutions, there has never been a meaningful gap between what hardware is capable of producing and what has been produced as output. More significant perhaps, is that in recent years the gap between interactively generated output and what we perceive in non-computer related media (or indeed in real life) has been appreciably reduced. The goal of realism has until recently been reserved for off-line approaches, but, with advancement in graphics capabilities, interest is increasingly being directed towards interactive graphics where solutions of close to the same quality are now required at real-time rates.

One such area is the physically based motion synthesis of objects in a virtual environment and the interactions of such objects when they collide. This is a historically difficult problem necessitating simplifications in order to manage computational complexity. It has become a sub-goal in such systems to adaptively apply simplifications to different parts of the scene, trading off complexity for processing speed wherever required. To optimise this trade-off, it is desirable that any loss or increase in detail is introduced seamlessly into the system and some approaches, such as the one described in this thesis, achieve this by taking into account the user's perception of the scene.

1.1.1 Physically Based Animation

Physically based animation concerns itself with simulating, in the virtual world, the same properties and behaviours of objects that we see in the real world. As opposed to traditional *ad hoc* or artistic animation techniques, simulation entails enforcing known rules and constraints that will govern the motion of objects in the animated scene. To illustrate this distinction, we might take the example of an animation of a rubber ball being thrown on to the floor. Both approaches to animation should of course end up with consecutive graphical frames, where we see the ball at different positions in the scene, possibly falling first and then bouncing up off the ground. In the non-physically based approach, the animator is left to decide how fast the ball should move at each frame, when it should start moving back up and how far it should bounce. In contrast, the physically based approach requires the animator to specify the starting size, speed and position of the ball, some physical properties of the colliding object *e.g.* mass and coefficients of restitution, and possibly global properties of the scene such as gravity and other forces that might act on the ball. Once these variables have been input, the animation system is solely responsible for generating the animation frames where the ball behaves in expected ways, such as colliding, falling and bouncing.

In essence, physically based animation is an extension of procedural animation[1] with the additional prerequisite of using laws of physics to drive key procedures that generate the animation sequence. Although some level of control can be lost by the animator in such procedural systems, physically based animation offers

[1]Procedural animation systems use programmable sets of rules that can be repeated or replayed as required to generate animation.

several advantages. Firstly, being based on physical laws, the generated motions are often much more realistic than can be generated by procedural animation or by a human animation artist. Provided that realism is our primary goal and provided that the quality of our physically based procedures and input variables are adequate, this is an important factor in choosing a physically based animation system. Secondly, there is the benefit of reusability and automation in the animation system as, once the system has been set-up to enforce the necessary physical laws, then different sets of input variables can be used to generate multitudes of different simulations with minimum effort. Furthermore, this kind of reusability is essential in interactive environments, where the users' actions may be used as input variables for the animation. Physically based animation makes it possible to generate realistic real-time animations on the fly, where the user is able to interact with the environment and manipulate the animation.

Unfortunately, the simulation process involves many complex and computationally expensive operations. As a result there is a big divide between the complexity of off-line simulations, with no significant time constraints for generating animations, and real-time simulations, which are required to generate and deliver animation frames at least twenty times a second or more often. Despite this fact, physically based real-time animation has become a hugely popular area in past years. As the processing power of the average home computer increases, we are seeing the potential to deliver, in standard interactive applications, a complexity of physical simulation that a few years ago was reserved only for the off-line animation industry and for commercial installations using high specification and high cost hardware. However, as all simulation concerns itself with modelling a continuous and infinitely complex world, we must accept that for many years to come we will have to be content with a simplified discretisation of our theoretical ideals.

1.1.2 Adaptive Simplification

Fully accurate and highly complex simulation would seem to be an unattainable goal and early approaches have simply dealt with this fact by simplifying the subjects that they model, restricting themselves to a world that will only ever get as complex as they think they can handle in the worst case scenarios. However, such pre-emptive simplification suffers from a few obvious flaws. Firstly, due to the nature of the dynamics system, which by very definition is ever-changing, it is difficult if not impossible to predict how complex any specific scenario is going to become. Computational complexity in such systems arises not from the number of objects or features but from possible future states of objects in the system. For instance, whether or not there will be many simultaneously occurring collisions during the simulation will affect how much processing power will be required. While it may sometimes be possible to predict from initial conditions what future states will arise, the introduction of an external random variable, such as human user input will usually prevent this.

Even if we were able to predict all future scenarios and restrict the complexity of the system, we often find that the worst case scenarios occur relatively infrequently and furthermore that the computational workload for these situations can be grossly higher than the average case. This means that programming for the worst case invariably results in simulations being over-simplified and processing power wasted for the sake of a few snapshots of high complexity.

Pre-emptive simplification, although simple enough to implement, is therefore not the ideal approach to take for long-running simulations of potentially high complexity. It would be ideal if we could instead adaptively simplify the complexity of the ongoing computations at run time as the simulation takes place, thus reducing complexity only when necessary and where it is most needed.

The process of simplifying a simulation might be achieved by simply switching on or off the processes that drive the simulation for objects that are flagged as being less important in the system. Alternatively, we could use different processes of varying complexity and accuracy to drive different parts of the simulation. For instance, Carlson and Hodgins [13] used different simulation algorithms to animate homogeneous objects in different parts of the scene, using the most complex and realistic model for dynamics in the most important part of the scene. Although such an approach might be termed "adaptive", because the simulation detail is modulated for different regions, it still lacks some flexibility as the boundaries of each level of detail region are predetermined. Although the overall simulation will tend to be more optimised, particularly if the regions are chosen well, the problem of worst case scenarios causing bottlenecks still remain. For instance, should all simulated objects for some reason crowd around the regions of highest importance we can end up with the same problems as an approach with no such levels of detail.

Time-critical approaches go one step further by taking into account a set time-schedule for delivering animation frames. The scene might be split up into similar regions of importance as in the previously described approach, with processing beginning by simulating all objects at the lowest level of detail. Then, as long as there is time available, a more accurate simulation model is applied to the more important objects. This stage is then repeated until either the highest level of simulation is reached or allocated time for processing runs out. The advantage is that if time should run out, there is the lower level of detail to fall back on. Provided there is time at least to simulate all objects at the lowest level of detail, it becomes possible to guarantee any given frame-rate requirement.

A further step to optimizing the trade-off between speed and accuracy is to use perceptual prioritisation to choose different levels of detail for the scene. In other words, objects or parts of the scene that were considered to be perceptually important would be given high priority and would be the first to be resolved at higher levels of detail. Perceptually adaptive compression has been used effectively in industrial applications such as JPEG compression and Mpeg-4 where Perceptual

Noise Substitution is used to compress audio. Perceptually based graphics is a fast-growing area as perceptual science is not only an effective basis for a strategy for simplification but also for evaluating existing non-adaptive techniques for rendering.

1.1.3 Collision Handling

Accurate modelling of collisions between objects in the scene is a vital feature of a physically based animation system. It is mainly through collisions that the state of one object can influence another object in the scene and it is through collisions that a user of a physically based virtual environment is able to interact with and affect the elements of the scene. Unfortunately, collision processing is an inherently expensive procedure as it introduces computational workload, not just in calculating correct physical responses, but also in detecting when these collisions occur in the virtual world in the first place.

As scene complexity increases, collision handling quickly becomes a major bottleneck in the simulation process. Thus, a trade-off between speed and accuracy in collision handling is particularly useful in order to achieve interactive frame-rates. Our goal is to optimise this trade-off by making simplifications as imperceptible as possible to the viewer in the animation of rigid bodies and articulated figures composed of rigid subparts.

1.2 Overview of this Thesis

1.2.1 Objectives

The goal of this thesis is to present a fully time-critical and perceptually adaptive approach to the collision handling problem in physically based real-time animation. Specifically, we wish to extend previously proposed interruptible collision detection systems to incorporate time-critical contact modelling and collision response. Established approaches optimise the collision detection process between potentially colliding objects by using bounding volume hierarchies to localise potential contact queries. However this type of approach has not hitherto been used for delivering physically based collision response. In this thesis, we propose an approach to using bounding volume intersection tests themselves to generate approximate contact data which can be used directly by a physics-based collision response system.

We will introduce the notion of approximate contacts, which can be progressively refined to deliver varying degrees of accuracy in collision response given increasing amounts of processing time, thus facilitating a system which is capable of delivering adaptive levels of detail in collision handling. All level of detail approaches bring about trade-off between speed and accuracy in the system and out of this arises the need and potential for optimisation. Thus, a discussion of adaptive collision handling will not be complete without a detailed study in optimising any required trade-offs in terms of computationally efficiency as well as the degree of plausibility of the simplified solutions. We will present details of some such optimisation techniques and provide evaluations of their effectiveness.

1.2.2 Scope

This thesis deals with the problem of collision handling of rigid bodies and articulated hierarchies of rigid bodies for use in interactive applications with strict frame-rate requirements. The overall accuracy of the system is sacrificed in order to achieve frame-rate requirements but this tradeoff is carefully controlled in order to optimise the viewers immersive experience of the scene. The approaches discussed here provide approximate solutions to the mathematical equations governing the motions of objects and are not intended for use in critical applications where it is important that the resultant final states of the objects needs to be calculated to the highest possible degree of accuracy. Instead, the solutions presented, attempt to maintain the instantaneous validity of the results and provide a real-time interactive user experience to the highest degree possible for any given scene. The solutions provided cater for scenes where objects are modelled as bounding volume hierarchies of a range of different types.

1.2.3 Contributions

We offer the following points as the specific contributions of the work that is presented in this thesis:

- We provide a thorough literature review of physically based modelling and the problems of collision handling in particular. We also provide detailed discussions of areas not traditionally associated with these areas. These are namely, level of detail modelling techniques, perceptual issues and plausibility.
- We present a general design philosophy for addressing the development of a system for real-time physically based animation by exploiting **adaptive level of detail techniques in collision handling**. We model pairwise collisions between simple nodes which are used as atomic components in a much larger system for constructing rich, high fidelity interactive virtual environments.
- We document the use of various **bounding volume hierarchies (BVH) for the specific purpose of level of detail collision detection**. Some new BVH schemes are discussed, namely the VoxelTree and the heterogeneous bounding volume trees.
- We also present an approach to **time-critical contact modelling and response**. Traditionally, approaches employing bounding volume hierarchies do so only for early culling, relying on accurate collision detection to provide contact manifolds and proper response. Our approach generates an

approximate contact model and collision response which is progressively refinable and time-critical. Node intersection data in the collision detection stage is used to pre-schedule time for the generally non-refinable response calculations thus ensuring a time-critical solution as a whole.

- We implement and discuss an efficient and useful extension of the bounding volume approaches to enable **collision and self-collision detection for articulated objects**.

- We present results of various **perceptual experiments to test the plausibility of offered approximate solutions**. We offer **strategies for optimizing the speed-plausibility trade-off** by highlighting the key areas that we have found subjects to be most sensitive to when approximate solutions are used in generating an interactive physically based animation.

1.2.4 Summary of Chapters

This chapter discussed the motivations to this thesis. The remaining chapters are organized as follows:

Chapter 2 provides a detailed review of the related areas of research. We discuss previous solutions to the problem of collision handling and highlight some of the open problems that remain.

Chapter 3 is a discussion of the adaptive time-critical approach for collision handling, which extends a previous approach to sphere-tree collision detection by providing an approximate contact model for bounding volume collision detection. Information is then passed between the three processes of collision detection, contact modelling and collision response, all of which function under a single cohesive time scheduler.

Chapter 4 discusses additional details for the optimisation of approximate contact modelling, presenting approaches for efficiently reducing computational workload for collision response whilst preserving as much of the high fidelity data returned by the processes of collision detection and contact modelling. We also extend the framework of Chapter 3 to encompass different kinds of bounding volume hierarchies included VoxelTrees, SSV-trees and heterogeneous hierarchies.

Chapter 5 discusses the perceptual optimisation of the adaptive level of detail system for collision handling using different scheduling strategies and provides a study of perceptual plausibility issues. We also discuss problems and solutions unique to adaptive Level of Detail Simulation.

Chapter 6 presents final results from an interruptible collision handling system and discusses some perceptual experiments to evaluate certain aspects of the work presented in this thesis. We begin by presenting different resolutions of output from our system and then evaluate how important it is to have adaptive level of detail in order preserve the over-all perceptual impact of a simulation. We then compare various scheduling strategies described in chapter 5 and present results of experiments to evaluate the quality of a gaze-driven scheduler.

Chapter 7 summarises the topics discussed in the thesis and describes on-going extensions and future research directions.

Chapter 2

Related Work

This chapter provides a detailed review of related areas of research and delivers a background understanding of the concepts that will be built on in the rest of the thesis. We will outline the application areas where adaptive collision detection techniques are required and present an overview of existing solutions and problems that remain. We first discuss the area of physically based animation to understand the requirements of the problem domain and the variables that we need to deal with. We then discuss the problem of collision handling in particular, as this is the topic we will be dealing with through most of this thesis. Lastly, we discuss adaptive level of detail techniques, which have always been prevalent in interactive animation but not widely used until recently in dynamics simulation and collision handling.

2.1 Physically Based Animation

Physical accuracy of some form has always been the goal in realistic animation. Whether the animation frames are drawn by humans artists or by a computer, when the goal is to represent reality then the laws of physics must be taken into account, even if only informally. Even cartoon animation, with its own unique system of physical behaviour, obeys laws such as gravity, friction and collisions, which are loosely derived from physics.

Computer generated physically based animation evolved from early procedural animation systems and can be defined as the process of generating frames of animation using numerical solutions to the motions of objects based on laws of physics.

In fact, what we are principally concerned with is the kinematics and the dynamics of objects in the virtual world. Kinematics is the study of the motion of objects, including their positions, orientations, velocities and accelerations. Dynamics, on the other hand, deals with the forces and impulses that cause object motions. Many authors use the term *Dynamic Simulation* to describe a procedural animation based on the laws of dynamics.

Physics based animation has been a topic of interest for some years, when generating increasingly complex systems with realistic motions. Amongst the many application areas, popular early areas of research were in rigid body simulation [5, 38, 69], physically based modelling of articulated characters [1, 95] and animation of cloth [93] and flexible bodies [39, 87]. Many of these systems, though highly realistic, dealt only with pre-generated off-line animations and only recently, with the evolution of optimisation techniques and developments in computational capabilities of machines, has physically based animation been of general interest in real-time animations.

We will provide a background to the full process of dynamic simulation, highlighting the key processes in Section 2.1.1 and discussing the stages that lead into collision handling. We then discuss the issue of modelling objects for simulation in 2.1.2. Most of the methods used in this thesis are based on well established laws, obtained from theoretical derivations or empirical models well-documented elsewhere [25, 29, 35, 36, 50, 86]. This thesis aims to discuss not the derivations and backgrounds to these laws themselves but how they are used and optimised for real-time animation. We do, however, need a good knowledge of the variables required for dynamic simulation, in particular those involved in collisions of rigid bodies and this is discussed in Section 2.1.3. In section 2.1.4 the actual process of applying these variables in order to compute realistic motion is discussed.

2.1.1 The Simulation Pipeline

The process of dynamic simulation concerns itself with the synthesis of motion based on physical laws. The system for dynamic simulation can be seen as a sequence of processes in a pipeline. We will discuss each of these stages here to establish some common terminology and a basic framework for further discussions later on in this thesis. In practice, a strict pipeline is not always adequate as the optimised solutions are often obtained when the processes are interleaved and less independent.

The stages in a theoretical physically based animation system are as follows (also see Figure 2.1):

Modelling : similar to the modelling phase in rendering, this concerns the representation of the object in a data structure that is useful for the rest of the simulation pipeline. Objects' physical properties such as spatial occupancy, mass and density distribution are determined and stored.

Simulation : this involves changing the states of the objects over time or from one simulation frame to the next. This is based on the state of objects and any external forces acting on them. For instance, an object's position needs to be updated based on its velocity.

Collision Detection : an object's free motion due to constant forces alone is not enough in a world of solid bodies. We need to check if objects in the scene are interpenetrating due to the simulation state changes.

Contact Modelling : once we have found colliding objects, we will usually require some more detail on the nature of the contact, *e.g.* which parts of the objects are interpenetrating or colliding. A *collision manifold* is the spatial representation of the contact region between two objects, which needs to determined before the next phase.

Collision Response : we need to enforce solidity by eliminating interpenetrations as well as applying new forces or impulses to objects based on how they have collided. Sometimes it may be required that we step back in simulation time to do this properly.

Rendering : once a valid and stable set of states for all dynamic objects is achieved, we may display the current scene.

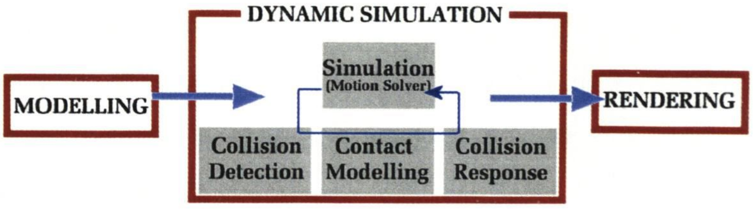


Figure 2.1: Simulation Pipeline

2.1.2 Modelling Objects for Simulation

The modelling of objects for simulation is in some aspects related to modelling for rendering. Similar data structures such as polygonal meshes and voxel arrays are sometimes used for representing the boundary surfaces and spatial occupancy of an object. However, the two modelling processes have quite different goals. Physical integrity of meshes, for instance, is usually more crucial in modelling for simulation while high detail is not as crucial. A polygonal mesh with small holes might be adequate for rendering but would likely cause unacceptable problems for the simulator. Conversely, a highly tesselated mesh might be important for a good rendering but may not be as useful for simulation. In fact, many approaches make use of the concept of a *proxy object*, which is a coarser representation than the display object used in rendering. As a simple example, in the modelling of a golf ball, we might use a detailed description of the surface complete with indented grooves for display, but replace the object with the proxy object of a sphere as far as the simulator is concerned (see Figure 2.2).

Generally, two often conflicting goals govern the choice of proxy object. On one hand it is important to choose an object representation for the proxy object so that dynamics computations can be performed as quickly as possible. On the other hand, an overly coarse approximation can cause visible inconsistencies when it comes to simulating the object during the animation process. Furthermore, we must take care that inconsistencies arising due to approximation do not cause errors that will threaten the robustness of the simulator. It is often good practice to take a conservative overestimate when generating the proxy model for an object, such as a convex hull or a bounding volume for the object.

As the physical properties of an object are more important, volume based methods are often more prevalent in simulation literature than in rendering. This is because boundary representation alone may not provide sufficient information for the simulator to generate accurate physical behaviours for objects. It is however possible to get accurate results with such models if we can assume that the object is a uniform density solid and if the boundary representation is closed. Using this assumption, many popular rigid body simulation applications such as *Havok* [1] achieve very good results by simply using parametric or implicit surfaces or a valid wire-frame mesh of adequate resolution as the proxy models for simulation. Many other systems use approaches that can be best described as spatial occupancy representations either exclusively or in conjunction with polygonal models [46, 47, 45, 43, 48, 74, 76].

[1] For further information see http://www.havok.com

2.1.3 Dynamics Variables

Apart from the volumetric description of an object, we also require certain common dynamics variables used in physics calculations. As we stated in Chapter 1, we deal primarily with rigid bodies or hinged hierarchies of rigid sub-objects in this thesis. However, many of the same variables are also required for soft object simulations:

The basic dynamic state of a rigid body can be represented in terms of:

- centre of mass
- position and orientation
- linear and angular velocity
- force and torque
- mass
- inertial tensor

Some of these, such as mass, are constants for the object while others are evaluated based on other properties of the object and the environment. Baraff [96] concisely represents all the variable parts of the dynamic data structure in a rigid body simulation state **Y**, based on position $x(t)$, orientation $R(t)$, and the total linear and angular momenta, $L(t)$ and $P(t)$ respectively:

$$\mathbf{Y}(t) = \begin{pmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{pmatrix} \tag{2.1}$$

Thus, a full description of a rigid body, including its temporally variable state, consists of its spatial properties as well as its dynamic variables. We will also generalise by assuming that a proxy object separate from the rendering model is

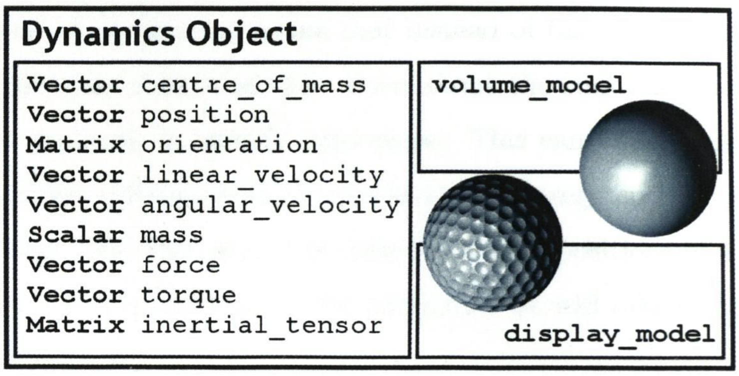


Figure 2.2: Dynamics Object Model

always used so that a full object representation can be stored as shown in Figure 2.2.

2.1.4 Simulation

Given a rigid body description and state as defined in the previous section, the process of simulation involves determining the change in the state variables over time. Primarily this involves integrating the equations of motion and, based on the object's properties and the forces acting upon it in one simulation frame, to calculate the change in state of the object from one frame to the next , *i.e.* by integrating $\frac{d}{dt}\mathbf{Y}(t)$.

Although this represents the core of what is involved in physically based animation, there is still a considerable amount of processing required, beyond straightforward integration. Collisions between objects due to their own motions and to user interactions with the world cause simulation discontinuities, which require less predictable state changes that need to be handled before the straightforward process of simulation by integrating object states can be resumed.

A significant part of the problem stems from the fact that time and space in the simulation world need to be discretised in order to be handled by digital machines. Simplification due to discretisation is a problem in all areas of computer graphics but, in the case of simulation, means that instead of taking into account all the positions of dynamic objects as they move about the simulated world, we only consider their positions at specific time-steps. This can cause problems unique to simulation when relevant events occur in the time segments in between these chosen time-steps. For instance, if we take successive positions of a sphere fired towards a plane, it is possible and quite likely that we will miss the exact time of collision (see Figure 2.3).

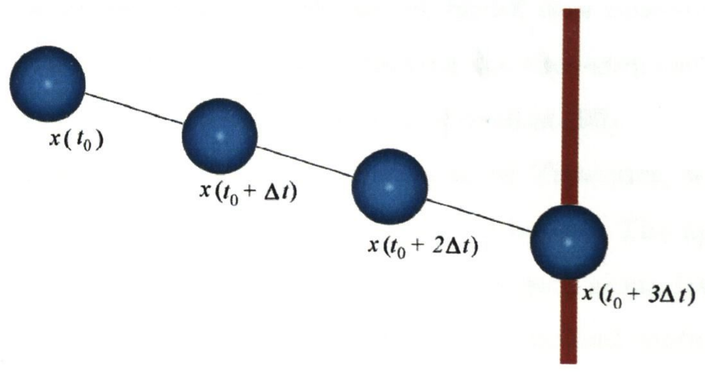


Figure 2.3: Interpenetrations instead of collisions due to discrete time-steps

In order to resolve interpenetrations, Mirtich describes three classes of strategies, implemented at the *motion synthesis* phase of the simulation pipeline [66]. One approach is to step back in simulation time to the time of first contact, respond to the collision by resolving interpenetrations and changing the objects' states and then continue the simulation from that point in time. This is referred to by Mirtich as *Retroactive Detection* and can be expensive since determining the point of first contact is not a trivial operation. Furthermore, the solution traditionally involves stepping back the entire simulation (*i.e.* all objects) to the time of first contact, which becomes infeasible for large numbers of objects.

An alternative method is *Conservative Advancement*, which involves finding a lower bound on the time between the previous frame and the time of first collision. If a collision is foreseen, a smaller simulation time-step is taken to find the intermediate position before contact. This is done iteratively until a contact time is found to some target degree of accuracy so that the simulation effectively "creeps up" to the time of first collision, as described by Mirtich. This approach suffers from excessive computational expense in calculating the lower bound time to first collision but reduces the computational wastage due to back-tracking. Once again due to the discretisation of simulation time, it is often impossible to reach the exact time of contact, so most approaches try to determine this within a certain tolerance level, ε.

A specific discretised solution presented by Baraff uses time-step bisection to iteratively step backwards and forwards, halving the time-step each time, until a solution under the specified error threshold is arrived at [96].

Mirtich provides his own solution, referred to as *Timewarp*, which performs back-stepping on independent groups of simulation bodies. The approach works by identifying dependency groups and back-stepping each group (rather than the whole scene) to the point of first collision for each group and continuing the simulation of each group from that point in time. This means that each independent group is resolved in its own separate discretised time-line. The different time-lines are then reintegrated at intervals of Global Virtual Time before global events are processed, such as rendering the full scene.

2.2 Collision Handling

Collisions between objects in a virtual world are an important feature in physically based animation. It is through collisions that interaction is made possible between the user and the scene and even between different objects within the scene. Collisions bring about discontinuities in the simulation process and are thus inherently problematic. Furthermore the three processes of collision handling, namely collision detection, contact modelling and collision response, are computationally expensive problems to solve accurately.

2.2.1 Collision Detection

Collision Detection has been a topic of interest for some years and obviously some form of collision detection is necessary even for the most basic two dimensional application. Pixel or sprite-based collision detection has been used since the very first interactive graphics applications were developed [22] and interestingly, very recent work has dealt with hardware assisted image-space techniques which bring the problem back to some degree into the 2D domain [51].

In the 3D stage, much early research relates to path planning and robotics applications [10, 12, 80].

Currently the popular approaches to accurate Collision Detection fall into two broad classes, namely *feature-based* and *simplex-based* methods. Feature-based techniques are generally concerned with the interrelations between specific features on potentially colliding objects [21, 59, 60, 65, 77]. Features include vertices, edges and faces of polygons on a polygonally represented object. Simplex-based methods treat a polytope as the convex hull of a set of points and perform tests on subsets of these points which are themselves simplices [11, 24, 33].

Also widely used are what we will refer to in this thesis as *approximate algorithms*, which involve associating with each simulation object an inexact and in most cases over-estimated representation of its volume. A substitution of a virtual object by a simpler one for contact and dynamics in haptics research is referred to as a *proxy object* [83]. We borrow from this terminology as our associated simulation object will similarly be of a form that will be easier and more efficient to process with regards to collision detection computation or dynamics calculations. Generally, the acceptable choice is to use a proxy that fully encloses the object being modelled, referred to as a *bounding volume.* This is to ensure robustness, as visual and simulation errors caused by under-approximations are often more harmful than cases where the size of an object has been over-estimated. Furthermore, bounding volumes are often used as a coarse first stage of collision detection to eliminate trivial non-colliding cases before the more expensive accurate approaches are called upon.

Hybrid collision detection approaches involve a broad phase that implements fast time-saving trivial elimination techniques followed by a more expensive narrow phase, possibly involving one of the accurate collision detection techniques described previously [47, 53, 59, 75]. Sweep-and-Prune [21] is a broad phase collision detection approach that tests for overlaps of the 1-dimensional intervals created by projecting the objects' bounding volumes onto the x, y and z axes. Sometimes an intermediate phase, referred to as the progressive refinement phase, is called upon after the broad phase, where increasingly accurate versions of the proxy object are referred to for collision detection [72, 76]

Essentially collision detection involves determining if and when two (or more) animated objects, defined by their dynamics states, *i.e.* position, velocity, acceleration, collide within a certain interval in time. Mathematically this would be achieved by checking for intersections in the space-time occupancy curves of the potentially colliding objects. For complex objects with complex motions this is an inherently difficult problem to solve generically. Further complications arise in interactive applications as we are no longer dealing with single space-time graphs that can be determined based on the starting states of the objects. Due to the nature of computer animation and in particular real-time applications, we are required to look to the domain of discrete mathematics. Our interest therefore is in solving a similar problem for finite time frames and discretised space-time bounds, which is what most of the previously discussed approaches attempt to do in some form. Hubbard refers to this kind of problem as *Interactive Collision Detection* [46] and discusses three key weaknesses that can arise:

Fixed time-step weakness : As discussed in 2.1.4, discrete time is a cause of major inconsistencies in the behaviour of dynamic objects. This problem can be alleviated somewhat at the expense of computational efficiency by choosing a smaller time-step Δt, but the problem persists regardless if object motion between timesteps is not considered. A partial solution to this problem is the use of Conservative Advancement, Retroactive Detection or Timewarp, as discussed in Section 2.1.4, to attempt to reach a solution within a pre-specified error tolerance. More precise results can be obtained through the use of space-time bounds as discussed in Section 2.2.1.

All-pairs weakness : A naïve solution to detecting collisions in the scene is to check for intersection between every possible pair of objects. This is an $O(N^2)$ problem where N is the number of objects in the scene. For the type of scenes that are in demand today, N can be problematically large. Broad phase approaches, such as sweep and prune or space-time bounds, allow alleviation of this problem.

Pair-processing weakness : For pairs of objects that have a high possibility of colliding, the collision/intersection computations between two objects is not a trivial task. Added to this is the requirement in physically based systems of determining the exact properties of the collision *e.g.* contact points or surfaces. If the pairwise detection algorithm is not robust and efficient this causes problems of accuracy of output as well as over all stability in the system.

In this thesis, we offer no improvements to alleviate the all-pairs weakness, instead assuming that this will be taken care of in the broad phase by sweep and prune or an alternate broad phase technique, which we ensure is compatible with our collision detection approach. We will discuss space-time bounds for alleviating the fixed-timestep weakness, but our primary focus will be on an approach which targets the pair-processing weakness by using a progressive refinement method for collision detection as well as the ensuing next stage of contact modelling.

Space-time Bounds

Space-time bounds represent the occupancy in 4D space (time as the fourth dimension) of a dynamic object and are used as part of broad phase collision detection to speed up the finding of the earliest collision time. Intersection between the space time bounds of two objects over a certain time interval Δt indicates that the two objects may possibly collide during that time interval. Hubbard presents an approach which uses Parabolic Horns and Hypertrapezoids to model the space-time bounds of object to determine potentially colliding objects in order to address the fixed-timestep weakness by adapting the time-step as required [44, 47, 45]. Cameron generates space-time bounds by four dimensional extrusions based on the dynamic states of objects [9]. Simple space-time bound intersection data is also used for generating dynamic contact groups which are use for adaptive simulation in Timewarp [66].

Bounding Volume Hierarchies

A bounding volume for an object is a geometrical entity whose volume complete bounds the actual object's volume. Bounding volumes are useful as conservative over-approximations of objects' spatial occupancies in calculating intersection of the objects for visibility or collisions. Bounding volumes usually take the form of primitives such as spheres and cuboids that are geometrically simpler than the objects they bound and thus are computationally easier to deal with in intersection calculations. Common practice is to first test an object's bounding volume and only perform the more expensive test with the underlying physical object if the bounding volume test has returned a positive result. Otherwise a negative result is returned, as a non-interpenetration with the larger bounding volume signifies with certainty that there is no penetration with the underlying object.

The converse however is not true and should the bounding test return positive, there is still a likelihood that the physical object test will return negative. In this

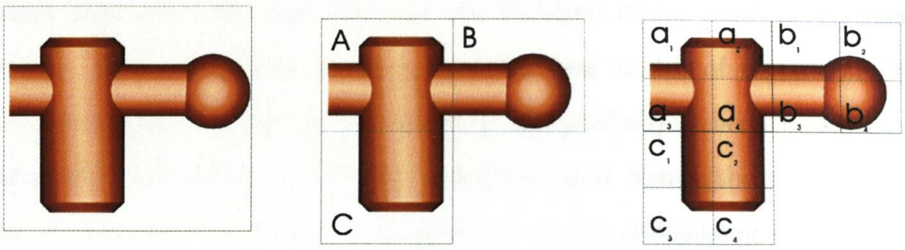


Figure 2.4: Bounding box and two levels of a bounding hierarchy.

case, there is wasted computation in performing the bounding volume test. It would therefore be prudent if we could increase the likelihood that the bounding volume result will be the same as the physical object test. This is automatically achieved by making sure that the bounding volume is as close a representation of the physical object as possible. This is sometimes not possible with a single geometrical primitive, so one approach is to use a collection of primitive bounding volumes to represent a single object.

A *bounding volume hierarchy* is a data structure that is highly optimised for intersection testing and consists of collections of bounding primitives approximating the physical object at various different levels of detail. Not only does each lower level of detail (*i.e.* looser approximation of the object) fully bound the next higher and thus more computationally expensive level of detail, but each individual bounding primitive encloses a subset of the primitives in the higher level. This means that in Figure 2.4 the nodes a_i are all children of A, and that A completely bounds all parts of the objects bounded by its children. Common bounding volume hierarchies are sphere-trees [7, 47, 76], axis aligned bounding boxes (AABB) [23], oriented bounding box (OBB)[37] trees, and Octrees [40].

Typically, such data structures are stored in a hierarchical tree structure, with each level of detail stored as a single level of depth in the tree and each node in the tree bounding its children nodes. Intersection testing begins at the coarsest level, the root of the hierarchy, which is usually a single bounding volume. If the intersection test fails then no more work needs to be done. If, however, an intersection is found with this level, all the children nodes must be checked for intersection. This procedure recurses until the leaf nodes of the hierarchy are checked revealing for certain if a test with the physical object is required. In some approaches the leaf nodes are actually primitives that define the real object, *e.g.* polygons on the mesh of the physical object or voxels composing a solid model.

This optimises computation by localising the parts of the object that are tested to a high degree of accuracy, thus, effectively addressing the pair-processing weakness discussed earlier.

The hierarchical data structure also forms the basis for performing what is termed *graceful degradation* of collision handling [26, 47]. As the different depths in the tree can be used to model the object at different levels of detail, it is possible to perform collision detection and contact modelling calculations at different degrees of accuracy. Each successive level offers more accuracy at the cost of a higher computational workload, thus enabling an important speed-accuracy trade-off. Initially calculating collision data at the lowest (and cheapest) level it is possible to progressively improve the accuracy by inspecting the next level of resolution and repeating as long as too much time has not been spent in this process to prohibit the achievement of a certain frame-rate. Time-critical approaches carefully monitor the amount of time spent during this process and interrupt the iterative refinement process when some set limit is reached. If the system happens to be in the middle of calculating the next higher level of accuracy at this time, then it is possible to simply fall back one level of accuracy and still deliver the final result in the target frame time.

Many different schemes exist for modelling objects using bounding volume hierarchies, the goal always being to balance a trade-off between speed of computation in performing intersection tests and tight boundedness of the bounding hierarchy. In choosing a bounding volume scheme for practical applications, we must also consider the difficulties in generating BVH approximations of objects. Important factors to consider include:

branching factor of bounding tree : BSP-trees, quadtrees and octrees use constant maximal branching factors throughout the tree making it easier to manage storage of data. However, this is not the case for all bounding hierarchies. Having a large branching factor or a variable branching factor can not only increase storage and computational complexity for collision detection but also causes extra complications in the automatic generation of the bounding volume tree.

atomic node type : Geometrically simple node types provide advantages in ease of storage as well as in updating node positions and performing collision detection. The trade-off here is mainly between ease of collision detection tests involving the node and the tight-boundedness of the node. Essentially, tight-boundedness determines how close the collision detection data due to the approximation matches what would happen with the actual object. Ultimately, this has consequences in how useful the data that is returned by the system will be, no matter how quick or slow the computation process.

uniformity of nodes : Traditional bounding volume hierarchies use homogeneous nodes throughout the tree. Octree, Octree-based sphere-trees[40, 76] and Vox-trees impose homogeneity restrictions on the dimensions of the nodes in each level of the tree. Others, such as AABB-trees[23], OBB-trees[37] and medial-axis based sphere-trees [8, 48], allow for some variance in the size and shape of primitive nodes, while yet others allow for characteristically different types of nodes (*e.g.* SSV-trees [55, 56], kDOP hierarchies[54] and C-trees [97]). It is however feasible and sometimes advantageous to use wholly heterogeneous trees for modelling objects for collision detection, although the automated generation of the tree becomes increasingly difficult.

2.2.2 Contact Modelling

Determining whether or not two object are colliding is, in itself, a useful process for many applications such as path planning or user interaction. Therefore many approaches are satisfied with the results of contact determination, which simply returns a result of "colliding"/"not colliding". Alternatively, some approaches perform proximity queries to determine the smallest separation distance between two objects [28, 42].

Other approaches have robustness in mind and return a measure of penetration depth [52]. Usually this is so that the intersecting objects can be moved back to a safe (non-intersecting) state, sometimes referred to as the minimum translational distance required to achieve non-interpenetration.

In order for a physically based animation system to deliver correctly simulated behaviours, often what is required is a clear description of the points or surfaces in contact. Contact modelling is the process of defining, in terms and variables useful to the dynamics system, the various properties of a collision that might have a bearing on the manner in which it behaves in response to a collision. The exact nature of the required output is naturally dependent on the system used for collision response but typically involves some description of the collision manifold, which is a representation of the common contact area between two objects.

Previous approaches have delivered highly accurate results, mostly by solving the problem at polygon level for colliding primitives. As most of these already perform highly detailed collision detection tests, it is a logical next step to return equally precise contact manifold information for collision response by inspecting contacts occurring at the polygon level [59, 65].

Generally, bounding volume approaches are utilised in collision detection for optimising the intermediate phases and to reduce processing for the accurate final stages involving polygon level collision detection and contact modelling. However, in an interruptible collision handling approach, collision detection needs to cease immediately when the time quota runs out and contact modelling needs to take place. Hubbard takes this approach with a time-critical sphere-tree collision detection system but only modelled very basic collision response, simply choosing to reverse the velocities of colliding objects [47]. No real contact modelling was performed as the approach did not attempt to generate a physically accurate response. To do so involves returning reasonably precise data on the points of contact and directions to apply forces.

In BVH collision detection, returning this precise data is far from trivial, because the bounding volumes over-estimate the object and in many cases a bounding volume collision does not even signify an actual contact. How then are we to generate contact points and directions needed for collision response, or should we be performing response at all? Unfortunately, in an interruptible system, we must treat any potentially colliding objects as if they had actually collided in order to maintain the robustness of the system. The problem of how to model approximate contact is dealt with in detail in chapters 3 and 4.

2.2.3 Collision Response

While collision detection and contact modelling are essentially problems of kinematics, collision response has more to do with the dynamics of objects in a simulation and involves computing the change in the state of an object resulting due to collisions [38, 69]. Ideally, the general solution involves correctly computing the forces resulting from the compression and decompression of objects due to contact with each other. Naturally, this is a prohibitively difficult problem for real-time interactive animation, which is why most real-time applications deal only with relatively elastic collisions between objects that are modelled as rigid. Real-time non-rigid body collision detection and response techniques do exist for very specific cases [63], but more general solutions are presently only suitable for off-line animation or for very much simplified scenes. In this thesis we will primarily be dealing with rigid and hinged body collision handling though many of the results are applicable to collision detection and simulation in general.

Rigid body collision response is based on the assumption that, for perfectly rigid bodies, the collision time (*i.e.* the time between compression and decompression due to colliding contact) is infinitesimal. In other words, the forces acting on the two colliding objects are almost instantaneous and the net effect of the collision can be approximated quite accurately as an instantaneous impulse. The dynamic body states of two colliding rigid objects change instantaneously based on an impulse vector, $j\hat{\boldsymbol{n}}$, where $\hat{\boldsymbol{n}}$ is the direction of the impulse and j is the impulse scalar calculated based on the original states of the colliding bodies.

Once two objects have been found to be colliding, the contact modelling phase must provide the directions in which to apply the forces and the points where these forces are to be applied. Based on this information and the states of the objects, a reasonable collision response is determined by calculating the scalar j and applying an impulse to the two colliding rigid bodies resulting in a dynamic change of state. This is discussed in more detail in Chapter 3.

2.3 Simulation Levels of Detail

Despite the levels of computational power available, fully accurate physically based animation remains a challenge due to the fact that the physical world we are attempting to model is infinitely complex. Thus, no matter how precise our model, it will only be an approximation of the real world. This is not always disastrous as many simplifications in animation go unnoticed by the human viewer and traditionally the approach taken in computer graphics research is to attempt as much complexity as is visibly detectable. This visual uncertainty could in fact be exploited to simplify less obvious parts of the scene in order to reduce computational complexity. Barzel *et al* recommend striving for ***plausible*** rather than accurate simulation in animation [6]. In interactive rendering, Funkhouser and Séquin describe a method to achieve the "best" possible frames using adaptive methods, whilst preserving a fixed frame-rate [30] whilst Reddy suggests that in 3D graphics, we tend to generate far more detail than users can perceive [82].

Adaptive level of detail approaches to simulation problems trade accuracy for speed, simplifying where necessary, to meet the demands of real-time rates throughout the simulation. The problem of adaptive simplification is not a new one, nor is it limited to the domain of physically based animation. A broad range of approaches in computer graphics, use models or procedures of varying levels of accuracy to model a single object or phenomenon in order to optimise rendering time, storage space or transmission time. They achieve this by offering a lower resolution of output when resources are being taxed and/or when a lower resolution model might suffice to deliver the output expected by the user of the system. With regards to the modelling of objects, *multi-resolution* representations of the same object can be generated by using different numbers of basic primitives to model the same individual object *e.g.* triangles in a triangular mesh, or by using a different number of iterations when polygonizing a mathematically defined surface (see Figure 2.5).

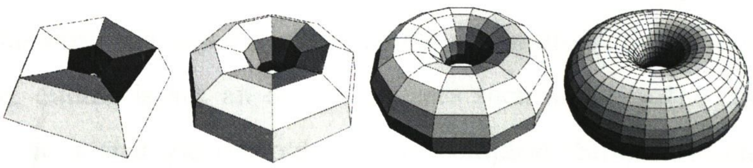


Figure 2.5: A torus defined at different levels of detail

Processing optimisation is even more important in interactive animation due to real-time frame-rate requirements, so Level of Detail techniques are required not only in modelling and rendering of objects but in synthesizing their motions and behaviours. *Simulation Levels of Detail* (SLOD) involves using processes and models of varying computational complexity in order to optimise speed-accuracy trade-offs in an attempt to achieve target frame-rates [70] .

Many physically based animation systems achieve real-time rates by culling computations for parts of the scene based on visibility. For instance, Chenney [14, 15, 19] reduces computational workload by culling dynamics computations from areas of the scene that are not visible but stress the importance in dynamics of paying heed to interdependencies between states of objects within and outside the visible scene. They highlight three factors that must be considered before culling dynamics, namely:

consistency : the state of an object when it returns into view needs to be consistent with its last known state.

completeness : Everything that should occur in the visible scene should still happen even when it is dependent on non-visible regions.

causality : events in the visible scene can affect non-visible regions. Even though we can not see them in the current scene, if these regions come into view later there should be evidence of these effects having occurred.

Thus, we must take care, in simulation, not to neglect even the non-visible parts of the scene, for as the simulation evolves or the view changes, dependencies often arise between the objects in the visible and non-visible parts of the scene. Furthermore, culling is not an option in cases such as when the visible part of the scene alone is still too computationally complex. Simplification within the visible area itself is required and a scheduling mechanism, which employs perceptual heuristics, may be used to determine which objects or events in the visible scene deserve more processing time. Most of these approaches are tailored to a certain type of scene or are intended to optimise a specific type of process used in the simulation. For example Carlson and Hodgins [13], and O'Sullivan *et al* [74, 75] discuss strategies for prioritising the scene for different types of dynamic simulation. Whilst, in the rendering level of detail literature, Funkhouser and Séquin [30] try to optimise interactive rendering for complex walkthrough scenes.

Reddy [81] discusses the problem in a more general sense, but the use of perception in interactive real-time graphics is still new and no known approach exists to date that attempts to address perceptual prioritisation in a scene in a completely generic *level of detail resolver*.

Once the scene has been partitioned into areas of varying priority, there are various ways in which we can decide where to make trade-offs between accuracy and speed. One approach is to use completely different simulation models, of varying levels of complexity, to evolve different parts of the scene as Carlson and Hodgins do for simulating the motions of one-legged animate creatures in a dynamics scene. Based on the importance of an individual simulated creature, different simulation processes were used, ranging from dynamics-based simulation to pre-calculated kinematics to point-mass behaviours for synthesising the creatures' motions. Chenney [19, 16, 17, 18] uses a discrete event simulator to generate coarse approximations of events that occur outside the higher priority region of the scene. Some approaches implement less distinct levels of detail to generate simulation levels of detail in animation by reusing single mechanisms with different resolutions of input data. These range from using adaptive time-steps for simulation or modulating the complexity of the proxy model used in simulation.

2.3.1 Perception of Dynamics

In the case of an interactive graphical simulation the quality of output that we achieve after any required simplifications and refinements can only be judged by its impact on the user. It can be said that the only reason we seek to simplify the scene by removing complexity in certain areas, is so that we can guarantee real-time frame-rates. In many cases, reductions in frame-rate can be more harmful to the user's experience of the virtual environment than any dynamic or geometrical reduction in accuracy, so the measure of how successful we have been in achieving the proper trade-offs in the system should really be dependent on the perceptual effect on the user. This suggests that perceptual impact is the measure that we should be striving to optimise as we allocate priorities to different parts of the scene.

Unfortunately, there are no established methods of objectively evaluating the perceptual quality of a dynamic scene. Therefore, it is difficult to obtain a measure of quality for a simulation without explicitly asking the user for his impressions after the simulation has been run. Not only is this very subjective but, since we deal with dynamic animations, meaning that it is likely that specific events will never be repeated quite the same way twice, we can not trust that the evaluation of one simulation run will have direct relevance to another, even if it has been executed with identical simplification techniques.

However, if we could better understand what lower level aspects of dynamic systems users are particularly sensitive to, this might lead us to better strategies, both in terms of prioritising a scene as well as in evaluating the effectiveness of a simplified scene. For this, we turn to previous literature on perception of dynamics to identify some key factors relevant to computer simulated dynamic scenes.

Causality

Causality refers to the ability to detect whether one event causes another. For example, a collision of a moving object with a stationary one will cause the second object to move, whereas if the stationary object only starts to move off by itself after a certain time period has elapsed since coming into contact with the moving object, it is perceived to be autonomous. The human brain appears to have very low-level processes that allow people to distinguish between animate and inanimate objects. Michotte showed that, even with such abstract objects, the impression that participants had of one event causing another or not was extremely strong [64]. In a related study, Leslie and Keeble [57] illustrated that six-month-old children could distinguish between causal and non-causal scenarios, showing that this ability is established very early on in life.

This is a factor that is highly relevant to a physically based animation system because it would suggest that the prompt processing of collisions is therefore necessary in order to maintain the perception of causality for collision events.

Separation

As we will see in following chapters, one of the possible consequences of reducing the accuracy of collision detection will be objects that bounce off each other at a distance. The extent to which this separation, or gap, between two colliding objects is perceivable will be an important factor in determining the ability of humans to detect an anomalous collision. This is because there is a topology-preserving mapping from the cells in the retina to the cells in the primary visual cortex, called a retinotopic mapping, that is quite precise and enables spatial location information to be efficiently processed [88].

Eccentricity

The fact has long been established in vision literature that many visual processing tasks deteriorate at increasing eccentricities from the fixation point [2, 94]. This eccentricity effect can be exploited in a real-time application by tracking the user's fixation position. Gaze-directed adaptive rendering has been investigated [62, 71, 91] and similar strategies can also be applied to simulation. When the viewer is looking directly at a collision, it would be given a higher priority than a collision occurring at a slight eccentricity, which itself would receive a higher priority than other collisions presented more peripherally. However, it may be that an eccentricity metric alone is not sufficient to guide adaptive processing in all cases and sometimes it is important to consider a combination of factors at once [2, 85]. Reddy discusses how a combination of velocity and eccentricity taken in combination can have a much greater impact on user perception of graphical elements in an animated virtual environment [81]. Although Reddy deals with visual perception objects, it is likely that similar results should apply to the perception of synthesized motions.

Multi-dimensionality

There is evidence to show that the human perceptual system relies heavily on representations of dynamic properties of the world. In a study of physics students taking a university course, Clement [20] reported that most had intuitive preconceptions concerning mechanical events that, although incorrect according to Newtonian mechanics, were highly stable and widespread. Proffitt and Gilden [78] showed that people use only one dimension of information when making dynamical judgements. Therefore, when a dynamic event involves more than one dimension of information such as velocity and rotation, *i.e.* an extended body motion as opposed to a particle that has only one dimension of information, humans are less able to correctly identify anomalous physical behaviour. It would be highly desirable if we could exploit this imprecision of the human brain by simplifying the scene in areas of high dimensionality.

Distractors

The presence of distractors is also likely to affect the ability to accurately detect collision or non-collision, as is the type of distractor. These issues arise in the area of Visual Search and have been investigated by researchers such as Saarinen [84] and Treisman [89]. It was found that if the distractors are perceptually similar to target objects that are being inspected, *e.g* have similar colours or common movements, then they automatically become placed in the same perceptual grouping and it becomes difficult to separate individual characteristics of the target objects from the distractors. This is however not the case when the distractors are significantly different, in which case the number of the distractors present has very little effect on users' abilities to distinguish their behaviours. O'Sullivan investigated the effect of distractors on users' abilities to detect anomalies in the specific case of collision detection in a dynamic scene [75].

User perception is discussed further in Chapter 5 in the context of developing good scheduling strategies for an interactive scheduler which prioritises collision events in the scene. In Chapter 6 we also discuss perceptual factors used as a metric for evaluating the plausibility of simulated dynamic events and also present results of some user tests to evaluate the effectiveness of a collision scheduling strategy driven by perceptual factors.

2.4 Summary

This chapter provided a detailed survey of the related areas of research, discussing some of the background to the problems faced and solutions that have been offered. We have outlined the need for optimisation in interactive simulation and the key sub-problem of collision handling, which we will be addressing in this thesis. We have described some previous strategies that have been undertaken in simplification for simulation, some of which will be directly relevant to our specific goals of collision handling and we have discussed some background to studies in perception that we intend to use for the purposes of evaluating dynamic simulations, as well as for strategies in scheduling collision handling in an adaptive level of detail system.

Chapter 3

Refinable Collision Handling

Collision handling is a particular area of physically based animation where there is great potential to save on computation time by optimising the speed-accuracy trade-off. Collision detection has long been a major bottleneck in physically based animation and it would be highly desirable to have a system capable of managing this difficult problem at different levels of detail. An interruptible collision detection mechanism adapts processing workload to fit a given time budget. However, such a system is not complete unless the data returned by it can immediately be of use to a collision response mechanism, which evolves the state of colliding objects based on how they have collided. This chapter describes a system that incorporates time-critical techniques into a full collision handler. The collision detection system here is primarily based on a previous approaches by Hubbard [43], which implemented time-critical collision detection but did not support any physically based collision response, which requires more than a simple contact determination query between two objects.

To ensure a completely time-critical system, we require that all processes of indeterminate complexity be packaged into the interruptible part of the system. Any calculations that have to be performed after that must take a constant or predictable time to complete. Alternatively, we can have several different mechanisms handling separate parts of the problem, each packaged in their own interruptible processes. Our time-critical approach to the collision handling problem uses interruption in the progressively refinable narrow phase to obtain approximate contact data. This must be interpreted by the contact modelling process before the collision can be resolved.

Up to now, contact modelling has been problematic, due to the inexact nature of even the most accurate of techniques available [65]. In interruptible systems, the problem is further exacerbated due to the reduced accuracy of results. A system that reduces computational complexity is ineffective if the resulting response is unbelievable. In an approach based on graceful degradation our first requirement is that the system delivers acceptably consistent results even with reduced input data from collision detection. Using perceptually based sorting it is possible to strategically simplify the scene and work towards the goal of plausible simulation, as discussed in following chapters, to exploit uncertainty and deliver real-time animation.

We describe contact modelling extensions to our own Hubbard-based system as well as a full framework to encompass the full process of collision handling under a single time-critical scheduler. The optimisations to contact modelling, in Chapter 4, and perceptually based scheduling, described in Chapter 5, are only made possible given that we have the interruptible system for collision levels of detail described in this chapter.

3.1 Collision Detection

The collision detection mechanism used in the system is based primarily on the approach described by Hubbard, which checks for intersections between the nodes of a sphere-tree data structure [48]. The key feature we wish to inherit from this approach is the hierarchical layout of the volume representation, which provides us with the basis for graceful degradation. We exploit the nature of the sphere-node, which provides us with a quick means of obtaining useful data for the collision response mechanism described later. Other useful advantages of this approach include an in-built method of handling collisions between concave objects, which other methods approach by using unions of convex objects; the sphere-tree by its very nature is already a union of spheres. We extend Hubbard's general approach with our own mechanisms for scheduling, contact modelling and collision response and we also generalise the approach, originally designed for sphere-trees, for use with different types of node-hierarchies (see Chapter 4). This is discussed in later sections but in this chapter we use the original sphere-tree implementation to illustrate the general principles behind the design of the framework.

3.1.1 The Sphere-tree

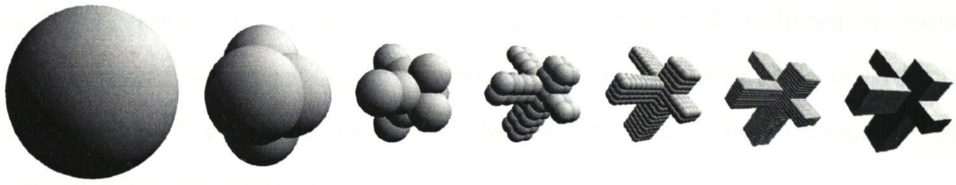


Figure 3.1: Sphere-tree Levels of Detail

Each object in the world is associated with its own sphere-tree, which is a spatial occupancy representation of the object's volume based on sphere primitives (see Figure 3.1). This is the proxy volume model that will be used for simulation. A sphere-tree is a standard bounding volume hierarchy (see Section 2.2.1) made up of sphere nodes. The sphere primitives are useful because collision detection between nodes effectively amounts to a distance test between the centres of the relevant spheres. A sphere-tree hierarchy is shown in Figure 3.2. Note, in the figure, the spheres highlighted in red which show the way that the hierarchy is structured with each parent node completely bounding the underlying volume that is then refined by its children.

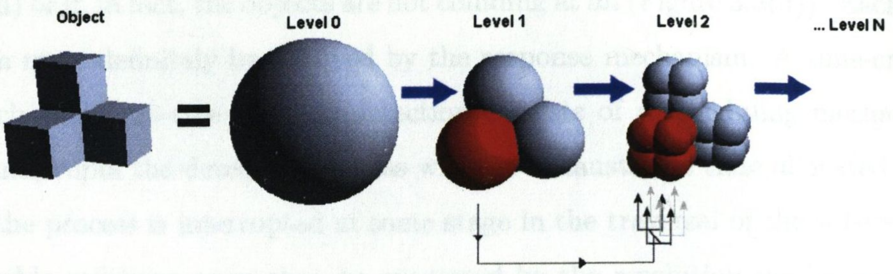


Figure 3.2: Sphere-tree hierarchy

The difficult part of such an approach, as with all hierarchical representations, is updating the hierarchy to reflect its change of state, which involves roughly one modelling matrix multiplication for every BVH node that is to be tested. We must update the position of each node to reflect the object's orientation before any intersection tests can be done. This can become computationally taxing as we go deeper down the hierarchy, but the workload is greatly reduced by updating only the relevant sub-trees *i.e.* the nodes for which the parent has been found to be intersecting. Some implementation details for sphere-tree collision detection are provided in Appendix B.

3.1.2 Sphere-tree Collision Detection

The collision detection mechanism is outlined in Figure 3.3. The process involves performing intersection tests for increasingly finer levels of sphere-tree detail. A possible collision is flagged when an intersection is found between sphere nodes on two different objects. Nodes requiring further processing are marked in bold in Figure 3.3 (a), (b) and (c). If the intersecting nodes are not leaves on the sphere-tree, then all of their children must in turn be checked for intersection. For this we follow a staircase traversal strategy as used by Palmer and Grimsdale [76]. Only when we reach the leaf nodes can we return a conclusive result as to whether the objects are colliding (in which case, we have an instance of a true collision) or if, in fact, the objects are not colliding at all (Figure 3.3(d)). Each true collision must definitely be resolved by the response mechanism. A time-critical approach to sphere-tree collision detection consists of a scheduling mechanism, which interrupts the detection process when it exhausts the time allocated to it. When the process is interrupted at some stage in the traversal of the sphere-tree, all possible collisions must then be processed by the resolution mechanism as if they were true collisions.

Some approaches, such as the one by Palmer and Grimsdale, use the progressive traversal of the sphere-tree as a middle phase and follow it up with a more detailed narrow phase consisting of polygonal intersection tests. Sphere-tree traversal in such an approach is used to localise the region to be checked later for polygon intersection. This requires further pre-processing time and data storage in determining the polygonal surface regions encompassed by each sphere, and also increases the time spent by collision detection in the more intensive polygon intersection tests. As in all polygonal collision detection approaches, there is the added the burden of collision response having to deal with several special cases of contact, depending on whether the contact involves combinations of faces, edges or vertices.

Our coarser intersection test will only yield a generic type of interference, that between two spheres. As we traverse deeper down our sphere-tree hierarchy, this approximation becomes increasingly more accurate. The radii of the sphere-tree nodes become smaller and their volumes more closely approximate the true surface. Progressive refinement is introduced into our system at the sphere-tree traversal stage. Thus, it constitutes the main part of the narrow phase in our system. We can compensate for the lack of polygonal detail by having deeper trees in our representation of the object than would be required of applications that depended wholly on narrow phase polygon testing (see Figure 3.4). This is in itself a significant trade-off, for if we could substitute the traversal of an added branch of the BVH tree by a single polygon test, there would be scenarios

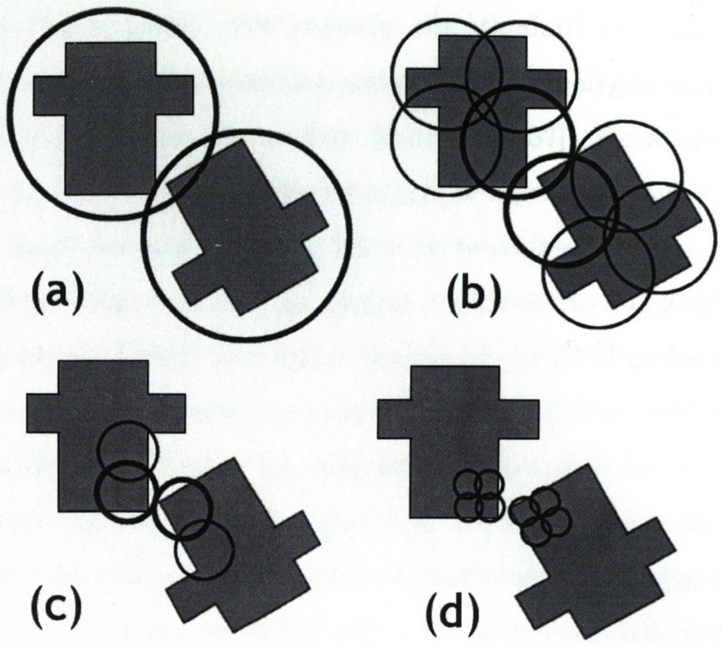


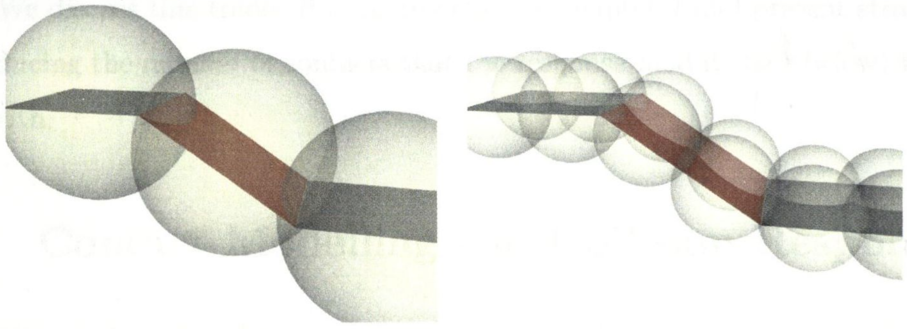
Figure 3.3: Sphere-tree Collision Detection: bold circles indicate colliding nodes which will be refined

where polygonal intersection could potentially return a quicker result. But, in our preference for having a fully generic system that propagates higher resolution tests of the same type, we will choose instead to deal with BVH trees that have been generated to resolutions comparable to the polygonal representation [1].

Figure 3.4 shows leaf node spheres of two typical sphere-trees. In approaches where BVH is used as a culling strategy for polygon level computations, it is common practice to have the finest level in the BVH hierarchy bound the polygons

[1]Several different optimisations exist for the polygonal intersection test. With most of these, processing time varies for different input variables due to the use of conditions for the quick rejection of certain trivial cases. As a result a thorough comparison of how spherical collision tests compare with polygon intersection tests is not provided here. In practice, however, it was found on average that, when only collision detection time was considered, approximately nine sphere-sphere collisions could be done in the same time it took for each polygon-polygon collision test. This easily supports the argument for having deeper sphere trees in favor of polygonal leaf nodes.

directly as in Figure 3.4(a). For instance, the standard approach to OBB-tree generation is to calculate a bounding cuboid for each polygon and build up the BVH hierarchy in a bottom up manner from these [37]. For a similar approach using sphere-trees, we can see that the polygon representation of the object is considerably more accurate than the spherical bounding volumes. Figure 3.4(b) shows a BVH tree that tries to approximate the model to one level of resolution higher. If we recurse further and reduce the size of the BVH nodes used to fit the object (and conversely increase the number of these), it is possible to get a close enough fit of the object so that the collision detection data converges to that for polygonal exact collision detection algorithms. Of course, the stage is eventually reached where it is computationally more efficient to perform polygon intersection tests instead of traversing a further branch of the tree but unless we have a means of dynamically proving such a case, the full BVH representation provides a much more generic implementation.



(a) (b)

Figure 3.4: BVH nodes approximating polygons

It must be remembered that our detection process is often interrupted at some stage during the traversal of the sphere-tree. To force the system to do further polygonal intersection tests, after the collision detection has exceeded the time allocated to it, defeats the purpose of the entire time-critical approach, as we cannot predict how long these will take. It is feasible to do polygon tests as the finest level of detail after the leaf nodes in the tree have been found to be intersecting. However, we must allow for the possibility that collision detection will be interrupted before our system has had the opportunity to do accurate polygon testing and that we will be required to compute a response based on whatever data we have gathered in the coarser intersection tests of the intermediate phase. Many others have dealt with polygon level collision detection [65, 11] and we offer nothing new in this regard. The principle concern in our system will be the handling of collisions at coarser levels of detail.

One side effect of having only the generic sphere-node collision is that certain edge and face collisions are sometimes detected as multiple sphere collisions, increasing the number of contacts that collision response has to deal with. The trade-off here is of multiple instances of a much simpler problem for fewer instances of a more difficult one (*i.e.* sphere collision detection as opposed to polygon collision detection), which is necessary in order for us to generically use the interruptible sphere-tree collision detection approach chosen in our implementation. We discuss this trade-off in more detail in Chapter 4 and present strategies for reducing the number of contacts that the response module (see below) has to deal with.

3.2 Contact Modelling and Collision Response

The primary goal of collision response is to deal with objects that the detection mechanism has identified as colliding. The response may be something as simple as changing the colours of the colliding objects, destroying one or all of them, or moving them apart so they are no longer colliding after the collision event. More is required in physically based animation, as the collision response mechanism has to calculate a change of state for the colliding objects based on laws of dynamics [69].

Solving the collision response problem can become a computationally intensive task if we take into account all the variables, such as elasticity, friction and energy, which contribute to the behaviour of colliding objects in a scene. There are countless mathematical laws in dynamics, which might be applicable to the problem of collision resolution. However, there is no general solution encompassing all these details that would be of use to real-time animation. So it is common practice to make some simplifying assumptions in order to provide a generically applicable system. Once again, we must sacrifice accuracy for speed and make the assumption of rigidity in order to provide a universally more applicable solution for interactive animation.

As explained in the previous section, our collision response mechanism has the added burden of dealing with approximated data, which must be selectively processed before it can be passed over to the part of the system that is responsible for the dynamics. We are chiefly concerned with the problem of using the refinable approximations returned by our collision detection system to obtain useful results for a real-time system. We will focus on a simple dynamics solver which deals with the problem of collision handling for rigid bodies. We model all of our colliding entities as perfectly rigid. Deformation during collision is infinitesimal and change of state is calculated based on instantaneous impulses as described by Giang [32], Baraff [4, 96] and Mirtich [67, 68]. We concentrate on the problem of contact forces, acting in directions normal to the surfaces of colliding objects, and do not handle the problem of friction forces during collisions. Mirtich showed how an impulse model can be applied to cases of resting and sliding contact as well as colliding contact.

3.2.1 The Dynamics Model

Working with an impulse model allows us to deal with the instantaneous values of the colliding objects' state variables. As we are dealing with approximations that change in their degree of accuracy from frame to frame, it is desirable that we are able to deal with the instantaneous values of the relevant parameters.

For rigid body collisions, a scalar j representing the magnitude of the impulse along the collision normal $\hat{n}$ is calculated using equation 3.1, which is based on [96]:

$$j = \frac{-(1+\epsilon)v_C}{\frac{1}{m_a} + \frac{1}{m_b} + \hat{\mathbf{n}} \cdot ((\mathbf{I}_a^{-1}(\mathbf{r}_a \times \hat{\mathbf{n}})) \times \mathbf{r}_a) + \hat{\mathbf{n}} \cdot ((\mathbf{I}_b^{-1}(\mathbf{r}_b \times \hat{\mathbf{n}})) \times \mathbf{r}_b)} \qquad (3.1)$$

The variables are

- ϵ: the coefficient of restitution, calculated from material properties of the two colliding objects
- m_i: mass of body i
- $\mathbf{I}_i$: the inertial tensor matrix of body i
- $\mathbf{r}_i$: a vector representing the displacement of the collision point $\mathbf{p}$ from the respective centres of mass $\mathbf{x}_i$ of the colliding objects as follows:

$$\mathbf{r}_i = \mathbf{p} - \mathbf{x}_i \qquad (3.2)$$

- $\mathbf{v}_C$: the collision velocity, which represents the relative velocities of the points of collision along the collision normal. This is calculated from the linear velocity $\mathbf{v}_i$, rotational velocity $\boldsymbol{\omega}_i$ and collision point displacement $\mathbf{r}_i$ of the colliding objects a and b as follows:

$$v_C = \hat{\mathbf{n}} \cdot ((\mathbf{v}_a + \boldsymbol{\omega}_a) \times \mathbf{r}_a)) - (\mathbf{v}_b + (\boldsymbol{\omega}_b \times \mathbf{r}_b))) \qquad (3.3)$$

Multiple Contact Points

When more than one contact point exists between two or more objects simultaneously in contact or penetrating, then the problem becomes one of calculating the net impulse acting on each dependent object based on Equation 3.1 for each object and three basic non-interpenetration constraints, namely:

$$\mathbf{v}_{rel}^{+} \geq -\epsilon \mathbf{v}_{rel}^{-} \tag{3.4}$$

$$j \geq 0 \tag{3.5}$$

$$(\mathbf{v}_{rel}^{+} + \epsilon \mathbf{v}_{rel}^{-})j = 0 \tag{3.6}$$

where $\mathbf{v}_{rel}^{+}$ and $\mathbf{v}_{rel}^{-}$ are the relative final velocities of two colliding objects.

The derivations of Equations 3.4, 3.5 and 3.6 are well detailed elsewhere as is the process of solving the Linear Complementarity Problem to resolve the individual net impulse magnitudes for all colliding objects [3, 4, 32]. We offer nothing new in this regard and merely state them here for completeness and to illustrate the workload that collision response has to deal with for each extra contact.

3.2.2 Approximate Contact Modelling

We discussed previously how the narrow phase of our collision detection system selectively traverses the sphere-tree to localise the region of interference between two objects. In the previous section, we identified the variables that we would require even in a minimal impulse based solution. Apart from the state variables of the individual objects, which should be directly available through their associated data structures, we need to know for each collision, a collision point, $\mathbf{p}$ and a vector representing the direction in which the required impulse is to be applied, $\hat{\mathbf{n}}$. Further processing is thus required before we have data of a form with which the dynamics solver is able to deal.

Exploiting once again the simple nature of our sphere-tree nodes, a quick approximation of the collision vector is the common normal to the two spheres which have been flagged as colliding. We can obtain this from the line that runs between their two centres $\mathbf{x}_a and \mathbf{x}_b$ according to Equation 3.7

$$\hat{\mathbf{n}} = \frac{\mathbf{x}_b - \mathbf{x}_a}{|\mathbf{x}_b - \mathbf{x}_a|} \tag{3.7}$$

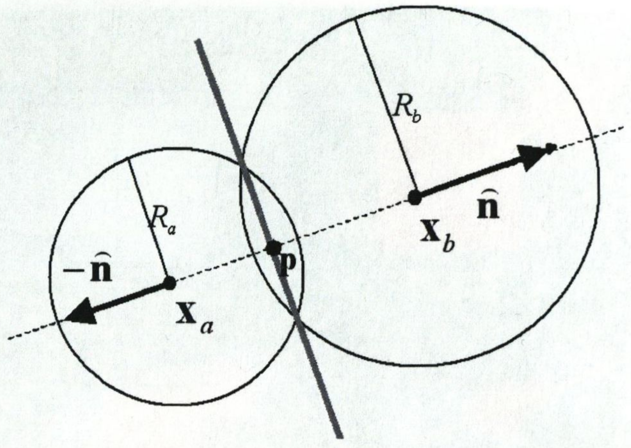


Figure 3.5: Determining collision point and normal for spherical node intersections

An approximation of a collision point is where this line crosses the plane of intersection between the two spheres (see Figure 3.5). It should be noted here that the sphere's in this diagram generically represent the nodes of the BVH hierarchy and thus this contact model applies to all progressive refinement levels of detail. The plane of intersection subdivides the line segment between the two centres into lengths proportional to their respective radii, and the point of intersection is determined by a simple proportionality calculation. For two colliding nodes a and b a common point of collision is required so $\mathbf{x}_a + \mathbf{r}_a = \mathbf{x}_b + \mathbf{r}_b = \mathbf{p}$, and:

$$\frac{|\mathbf{p} - \mathbf{x}_a|}{|\mathbf{p} - \mathbf{x}_b|} = \frac{R_a}{R_b} \tag{3.8}$$

This result can easily be calculated, even in the case where we are resolving intersections between spheres of different levels in the respective hierarchies of the two objects. Now that we have decided on a collision point and a vector along which to apply the collision impulse (Figure 3.6), we can apply the equations of Section 3.2.1 to generate the appropriate response.

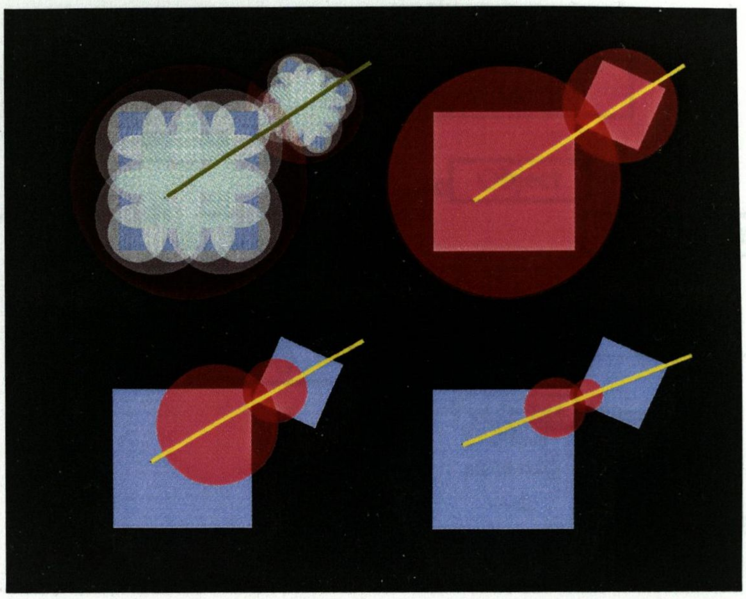


Figure 3.6: Collision Levels of Detail

3.3 Framework for BVH Collision Handling

A summary of the system design is shown in Figure 3.7. A **World** class encapsulates all the simulation concerns of the application, including collision handling and state propagation when there are no objects colliding. User interaction and all other interfaces are maintained by the **Application** class, which contains the world.

The key data structures in the application are the **Object** class, which stores the state of each body in our system; the **SphereTree** class; and a dynamically generated **Collision** list, through which the collision detection and response mechanisms communicate data.

The Object class needs to contain at least the state variables and constants shown in Figure 3.8.

A sphere-tree associated with each object is laid out in a hierarchical manner as already discussed in Section 2.2.1. Two instances of a sphere-tree are stored: the first ($body_S$) holds the untransformed body-space coordinates of the sphere nodes

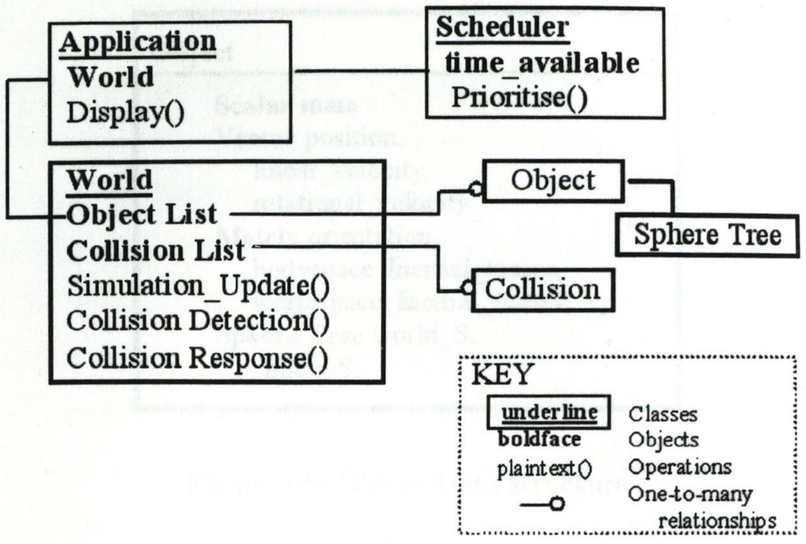


Figure 3.7: General Framework

and is constant throughout the simulation, while the second ($world_S$) represents the object in world coordinates after it has been updated with respect to the orientation and position of the body. Only the relevant nodes, *i.e.* the ones being processed by collision detection, are updated with the world transformation in order to save wasted computation time.

The dynamically generated collision list stores collision data in the form of collision entries, which are pair-wise descriptions of object collisions in the scene. Each collision stores more detailed data in the form of a list of **Sphere collision** data structures (Figure 3.9). The sphere collisions represent the specific intersections between nodes on the pair of colliding objects. This is the data that is required by the collision response mechanism. The world class carries out collision detection after every update of the simulation scene, generating the collision list of intersections at various levels of detail. Potential candidates for collision, in other words those objects not eliminated by the broad phase, are placed in a list of collisions for further processing. Starting at the root nodes we traverse the sphere-trees of the objects checking for intersection. When an intersection is

```
Object
    Scalar mass
    Vector position,
        linear_velocity,
        rotational_velocity
    Matrix orientation,
        bodyspace_Inertial_tensor,
        worldspace_Inertial_tensor;
    Sphere Tree world_S,
        body_S;
```

Figure 3.8: Object Data structure

found, the contact variables are calculated and an entry is stored in the sphere collision list of the respective collision object. Whenever we finish processing a complete level of a sphere-tree, we have the required data to refine the resulting response by one level of accuracy.

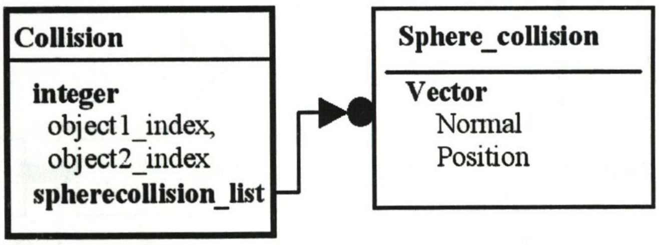


Figure 3.9: Collision Data Structures

3.3.1 Processing Collision Data

The final phase in collision handling is to call the dynamics solver mechanism which applies and resolves all the impulses to generate a mathematically consistent change of state in all colliding objects in the scene. At this stage, the contact modelling mechanism has provided us with all the variables we require, so computing and applying the required impulses for each instance of a collision is a fairly straightforward process of applying the equations of Section 3.2.1 to these variables.

All that is required now is to go through the dynamically generated collision table and to retrieve the relevant data. Several options present themselves as to how collision processing is to be organised. One option is to store collision data in a hierarchical tree structure similar to, but separate from, the way we have organised our volume data. A time-critical response resolution mechanism does a prioritised traversal of the collision tree, applying impulses at increasing levels of detail. In such an approach, we have the option of performing the contact modelling within the schedule of either the response or the detection mechanism. This approach conforms to the requirements we laid out for a fully time-critical system, with all processes of indeterminate complexity packaged in their own interruptible mechanisms (Figure 3.10).

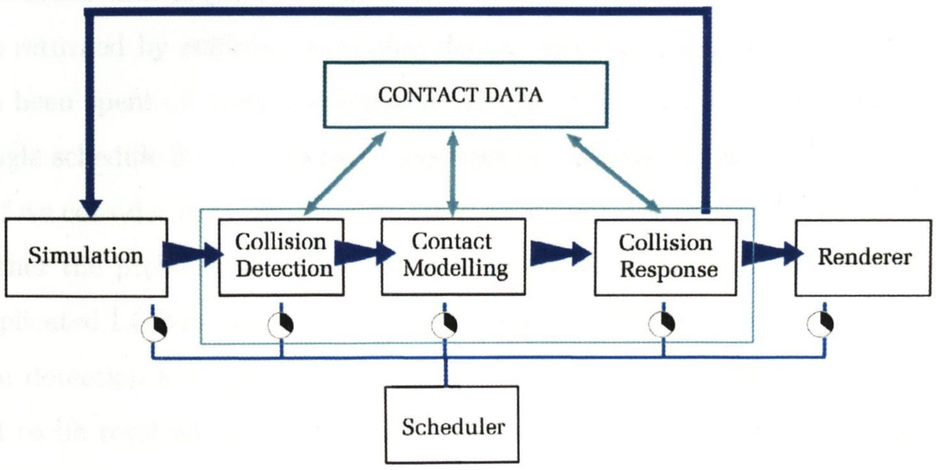


Figure 3.10: Scheduling of a full time-critical simulation system

However, it is often the case that data used in the collision detection phase is immediately of use to the response calculations. The two processes have many common requirements, such as the distance calculations and the data they return. Furthermore, in a system that schedules collision handling based on the visibility of the different parts of the scene, it is likely that we will wish to follow the same prioritisation scheme for both processes of collision handling. It would therefore be advantageous to have a single interleaved collision handling mechanism, with both the processes of detection and response working together and sharing data within the same schedule. Unfortunately the time taken by these two process as a whole is based on different sets of variables and thus it is not possible to predict how long each will take to complete. We can not assume that the two process will be of comparable efficiency, thus it is likely that having two separate schedules for two processes with such inter-dependency will lead to wasted scheduling. For instance, we may easily allocate too much time to the collision detection process and not be able to handle all the detailed data forwarded to the response mechanism before it is interrupted. The system remains time-critical but there is a lot of extra processing done by collision detection that is never used by response. Conversely, if too much time is granted in favour of response and it has dealt with all of the data returned by collision detection, we are wasting processing cycles that may have been spent on further collision detection. Thus it is more desirable to have a single schedule for the full three processes of collision handling.

If we consider that response calculations are of constant and predictable time, whether the problem involves a number of single contact collisions or the more complicated LCP resolution of multiple contacts, and that the time taken for collision detection is a direct function of the number of intersecting contacts that need to be resolved, we can see that it is possible to obtain a good estimate of the computation time involved with each extra contact point delivered. We can incorporate this into the time-critical scheduler by pre-allocating time for collision response based on the number of contact primitives that are being found by collision detection/contact modelling. In other words, not only is the scheduler's clock running down based on the time spent on current computations but we are also reducing the time-quota by pre-allocating it to collision response. This alone is enough to ensure fully time-critical collision handling, bundling all three phases into a single scheduler, guaranteeing any specified target frame-rate independent of the capabilities of the processor, provided that the system is capable of at least resolving all objects at the very lowest level of detail.

So, instead of having separate scheduling mechanisms for collision detection and response, a better alternative that fulfils the requirements of a time-critical system is as follows:

- The time-critical scheduler for the full application monitors the time spent in generating each frame of the animation. Taking into account the time required for other processes, such as rendering, it needs to decide how much time will be made available for the combined processes of collision detection, contact modelling and collision response.

- The scheduler initially allocates a time quota T_{frame} at each frame for all collision handling processes and decides on the order of collision processing. At each stage in the collision handling/contact modelling phase it not only checks if the quota has been exceeded by time already spent on collision detection and modelling t, but also reserves the time that will later be spent on the response computations $t_{response}$ as shown in Equation 3.9.

$$timeremaining = T_{frame} - (t + t_{response}) \tag{3.9}$$

- When it has determined that the detection process has exhausted its allocated time quota, it interrupts the collision detection process and passes control to the collision response mechanism. The response mechanism in turn updates the scene based on the most accurate level of detail available (in other words the most recently completed sphere collision lists for each collision).

This is illustrated in the revised collision handling pipeline in Figure 3.11 where we see that the collision response only reads from the contact data which stores only the highest level of resolution that can be achieved within the collision detection/contact modelling time schedule.

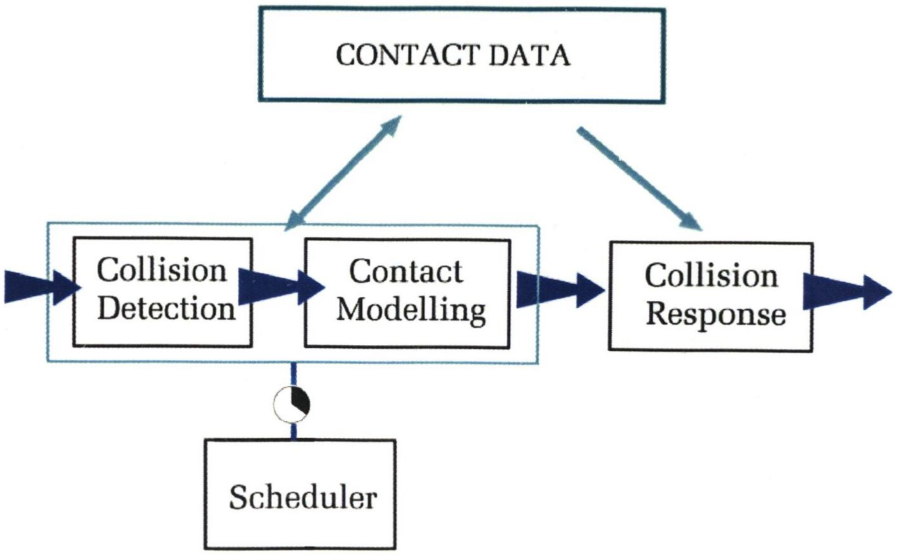


Figure 3.11: Scheduling of contact modelling and detection

3.4 Summary

This chapter has introduced a full framework for an interruptible collision handling system. Such a system makes it possible to achieve real-time interactive frame-rates regardless of the complexity of the scene, provided of course that the entire simulation can at least be handled in real-time if all objects are at the very lowest level of detail. The system is based on one previously presented by Hubbard, who did not, however, include the facility for realistic collision response within his framework. We have extended the system for full collision handling, including a scheduling process which takes computation time for physically based collision response into account. We will build on this in Chapter 4 by optimising the contact modelling phase in terms of performance and providing extensions in order that the framework will be equally effective when dealing with other classes of bounding volume hierarchies. We also optimise the trade-offs within the scene by using perceptually adaptive scheduling strategies for collision handling as described in Chapter 5.

Chapter 4

Optimising Contact Modelling

It is only due to the kinematic information returned by collision detection that interaction is made possible between the user and the virtual environment or between the objects within the virtual environment themselves. We discussed in Chapter 3 how an approximate contact model can deliver results that converge to a realistic collision response as more time is given to the system for processing the collision. Although the individual node contact is inexpensive to compute, BVH collision detection sometimes returns more contact points than exact methods. As a result, further processing may be required in order to reduce the workload for collision response. However, if by reducing the number of contact points the computed contact modelling data is not comparable with the proper contact details, then the ensuing behaviours will appear unrealistic to the viewer. This chapter discusses strategies for optimising the contact modelling phase, in terms of increasing the realism of the resultant output, as well as reducing computational load. We also discuss alternatives to the sphere-node contact model used as an example in Chapter 3.

4.1 Generic BVH Trees

In Chapter 3 we saw how sphere-tree data structures are used to deliver time-critical collision handling. Although the spherical nodes of a sphere-tree are useful due to their rotational invariance, there are many cases where they are not the most efficient choice for bounding volumes in terms of how tightly they fit an underlying object. For instance, in Figure 4.1(a), we see how bad spheres are at approximating flat shapes. Although the fit gets tighter with the next level of the sphere-tree (Figure 4.1(b)), a single box or rectangle swept sphere (see Section 4.1.3) gives a much tighter bound (Figure 4.1(c)). As we discussed earlier, tightness is important in approximate collision detection because tighter BVH nodes invariably produce results closer to exact collision detection methods. In this chapter we discuss a few alternatives to using spheres for the BVH tree.

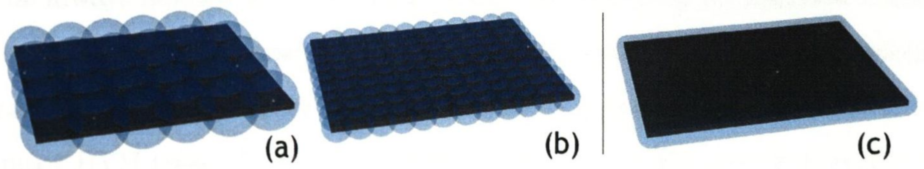


Figure 4.1: Bounding volume fitting for a flat rectangular shape

In order to fit any BVH scheme into our collision handling system, all we require is that the following be defined for the atomic nodes of the tree:

intersection model : procedures for collision detection, which should return whether two nodes are in contact.

contact model : procedures for contact modelling. This should return results compatible with our collision response system, which in our case consists of a contact position, an impulse direction and possibly a degree of interpenetration (discussed later).

4.1.1 Box based BVH Trees

We will define a Box as a six-sided volume where adjacent faces are orthogonal to each other and opposite faces are parallel. Next to spheres (or perhaps in certain cases in preference to them) boxes are the next most likely candidate to use as a nodal type for bounding volumes in contact determination. Although they are not rotationally invariant like spheres, they are a popular choice because of the efficiency and ease of intersection testing as well as the relative ease of generating the bounding volumes for objects. A simple 3D mapping from world-space to cuboid space reduces intersection testing to a simple matter of three one-dimensional overlap tests along the three Cartesian axes.

In fact, approaches such as Axis Aligned Bounding Boxes (AABB-Tree) [23] and Bucket-Tree [31] choose to impose the constraint that the bounding volumes must be always aligned with the world-axes, thus reducing intersection computations even further. Of course, this entails that the bounding volumes be calculated per-frame at run-time rather than at the pre-computation phase, as is the case with most BVH trees. Due to the relative ease with which this is possible (once again this involves simple one-dimensional comparisions to find minimum and maximum extents of an object along the three axes) and by exploiting frame-to-frame coherence, such approaches have been used to deliver good results and are particularly useful when there is a likelihood that pre-computed BVH descriptions will be invalidated, such as for deformable objects.

Axis Aligned Boxes aside, standard cuboidal trees are not only very efficient in terms of intersection tests but in terms of tightness of fit and storage efficiency. VoxTrees described in Section 4.1.2 and Octrees are efficient because of the regular arrangement and the homogeneity of the bounding volume nodes, meaning that positions (and orientations) of individual nodes need not be stored. This information is implicit in the description of the tree and can be calculated for individual nodes simply by examining their offset distances from (and orientations of) their siblings. In terms of tight-boundedness, perhaps one of the best bounding volume implementations is that of the Oriented Bounding Box Tree (OBB-tree) [37] which uses bounding cubes of heterogeneous orientation and dimensions to bound the object. Storage and intersection testing is marginally more expensive, with the added requirement of storing and applying the added inverse world transformation for each bounding volume node, as is tree-generation which is a fairly involved process. However, the variance in orientations and dimensions of the bounding nodes allows for much tighter bounds for a large range of different objects and as a result this approach is quite a popular one amongst the BVH Trees.

4.1.2 Vox Trees

A simple but useful type of box-based BVH Tree is a homogeneous cube based tree which we will refer to generically as a *Voxel Tree* or *VoxTree*. In this case, the term homogeneous will be taken to imply that the tree is made up of similarly oriented, similarly proportioned atomic nodes (regular cubes in this case) and we will also impose the constraint that all sibling nodes in the tree are of the same dimensions (Figure 4.2). A well known example of such a tree is the Octree which is an adaptive resolution tree of cubes with a typical branching factor of eight. However the following applies generically to Voxel based trees with arbitrary branching factors.

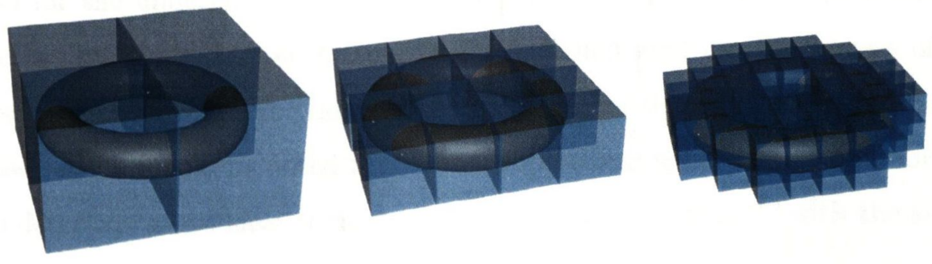


Figure 4.2: Three levels of a VoxTree

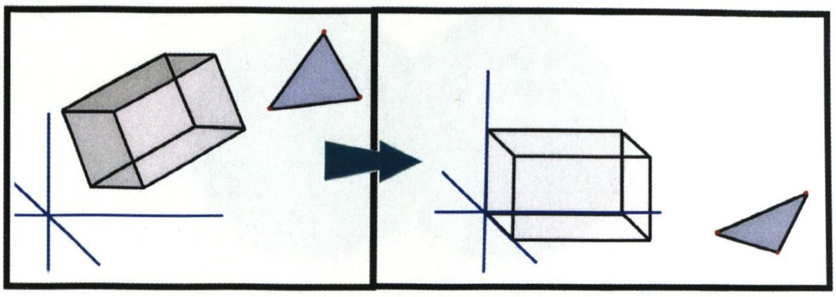


Figure 4.3: Mapping to cube-space

Intersection Test

Cubic collision detection is fairly straightforward. A number of approaches can be taken. We will use the method of mapping an object onto the cubes local coordinates and simply performing one-dimensional overlap tests with the object (see Figure 4.3).

Contact Model

Recall that the discrete collision response mechanism requires, for each contact instance, a point of contact, $\mathbf{p}$ and a normal, $\hat{n}$ for the impulse direction. For a quick approximate contact model we borrow from the spherical contact model effectively by circumscribing a sphere over the voxel (see Figure 4.4). This might seem like a drastic over-approximation at first since the cube is already an over-approximate bound for the underlying object. However, if we examine the spherical contact model we see that what we are primarily concerned with are the centres of the intersecting spheres and the ratios of their radii rather than the actual dimensions themselves. It should be noted that the circumscribing sphere is not used for collision detection as we have already achieved that in Algorithm ?? with the actual cubic dimensions of the voxel.

In most respects this is both an adequate and efficient contact model when the cube size is small in comparision to the actual object. Inaccuracies become more

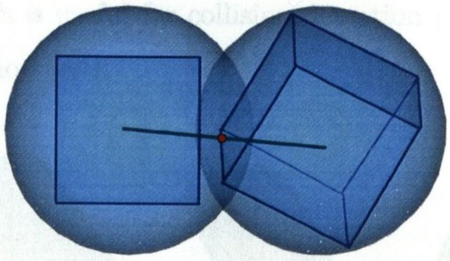


Figure 4.4: Contact Modelling for a Regular Cube

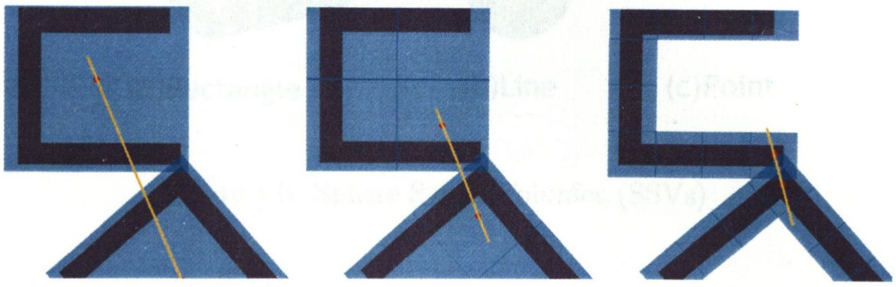


Figure 4.5: VoxTree Contact Modelling

evident should the system force contact resolution at the bounding cube level. In such a case the result is not ideal but is still comparable to the sphere model and overall the result converges to the accurate result with increasing processing time (see Figure 4.5).

4.1.3 SSV Trees

Sphere swept volumes (SSVs) are generated by taking the *Minkowski sums* of a sphere and a core primitive that can be either a point, a line segment or a rectangle to obtain respectively a Point Swept Sphere (PSS), Line Swept Sphere (LSS) or a Rectangle Swept Sphere (RSS) [55, 56] as shown in Figure 4.6. The Minkowski sum is the sum of point sets $\boldsymbol{A}$ and $\boldsymbol{B}$ in a vector space, equal to $\{\boldsymbol{a}+\boldsymbol{b} : \boldsymbol{a} \in \boldsymbol{A}, \boldsymbol{b} \in \boldsymbol{B}\}$. Effectively the above objects are simply obtained by taking the respective core primitives and extruding them by a certain radius as shown in figure 4.6.

The family of SSVs is useful for collision detection purposes due to the relatively simple intersection tests.

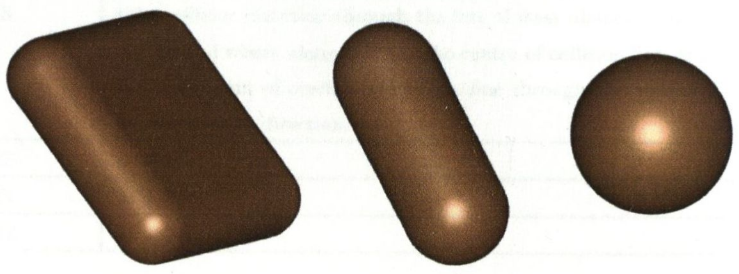


Figure 4.6: Sphere Swept Volumes (SSVs)

Intersection Test

Intersection tests basically involve a distance computation between the SSV's core primitive and the object being tested, *e.g.* for a point v.s. LSS intersection check, we test if the distance from the point to the LSS's axis is less than the radius of the LSS.

Contact Model

Slightly more complicated than the symmetrical cube and spherical contact model, the relevant pairwise contact models are shown in Table 4.1.

Usage

A hierarchy of Sphere Swept Volumes is ideal for hierarchically modelled virtual humans, as the volume nodes are a good fit for most of the nodes in the transformation hierarchies (see Figure 4.7). For characters modelled as a disjoint collection of rigid body parts animated based on hierarchical transforms, these transforms can easily be used to update the positions of nodes in the collision

PSS to PSS	as with sphere, collision normal $\hat{\mathbf{n}} = \frac{\mathbf{x}_b - \mathbf{x}_a}{\lvert\mathbf{x}_b - \mathbf{x}_a\rvert}$ and collision point is obtained from $\frac{\lvert\mathbf{p}-\mathbf{x}_a\rvert}{\lvert\mathbf{p}-\mathbf{x}_b\rvert} = \frac{R_a}{R_b}$
PSS to LSS	take collision direction through the line of least distance and apply $\frac{\lvert\mathbf{p}-\mathbf{x}_a\rvert}{\lvert\mathbf{p}-\mathbf{x}_b\rvert} = \frac{R_a}{R_b}$ to find where along this line the centre of collision lies.
LSS to LSS	take collision direction through the line of least distance and apply $\frac{\lvert\mathbf{p}-\mathbf{x}_a\rvert}{\lvert\mathbf{p}-\mathbf{x}_b\rvert} = \frac{R_a}{R_b}$ to find where along this line the centre of collision lies. If axes are parallel find the extent of overlap and take a line through the midpoint of this region as the collision direction.
PSS to RSS	as above
LSS to RSS	as above
RSS to RSS	as above

Table 4.1: SSV collisions

detection hierarchy as well. Thus, even though the macroscopic object is not rigid, we can still use our collision detection system because the individual nodes themselves are rigid and we can associate rigid bounding volumes with individual segments at the pre-processing stage when the object is modelled. A few small changes are required in the original system to support this functionality. Firstly, we must allow certain nodes in the tree to transform in relation to one another, unlike in the rigid hierarchy where a single transform is applied to **all** nodes. We should make a distinction here between the transformation hierarchy, which exists to enable animation using hierarchical transformations and the collision detection hierarchy, which exists to provide increasingly detailed collision modelling. Secondly, the top level bounding objects need to be more conservative than usual as they must bound their children nodes even after a deformation. For instance, in Figure 4.7 the movement of the arm and legs about the node centre should not be able to move any part of the body outside the topmost bounding volume. Unless we take this worst case approximation, it will be required that we recompute the dimensions of the parent node every time a child is updated.

For a node at the bottom of the transform hierarchy, *i.e.* a node with no children that moves in relation to its own position, it is possible to provide deeper levels of collision detection detail *e.g.* the head of the character might be modelled as a rigid object but it might still have children nodes to better model the features

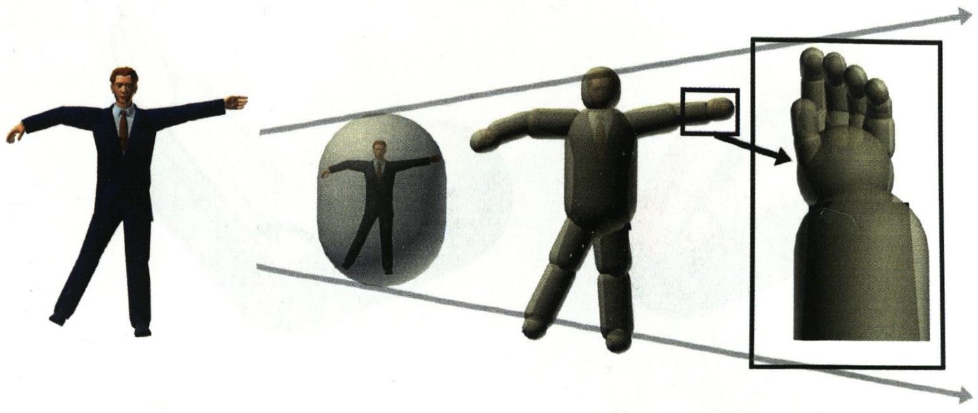


Figure 4.7: Line Swept Spheres (LSS) can be used to efficiently detect collisions with virtual humans.

of the head.

Within the transformation hierarchy, it is even possible to detect self collisions between individual nodes if we treat the individual segments as a soup of freely moving rigid objects. We make an exception between neighbouring nodes as these are usually made to overlap by design at the modelling phase. For these we can either choose to ignore all potential collisions or employ a technique based on deformable object modelling techniques [90, 79], which look at the normals of adjacent polygons to determine the possibility of intersection. In our case we look at the core axes of the adjacent LSS nodes and flag a collision if the angle between them falls below a certain forbidden threshold value $\boldsymbol{\theta}^{\mathbf{c}}$ shown in Figure 4.8.

The adaptive approach still switches between different levels of resolution of the character's volume hierarchy as required, thus facilitating the necessary level of collision detail. Rapid elimination of nodes from the collision detection phase is still made possible when their parent nodes are found not to be colliding. It is also possible to incorporate procedures for quickly detecting collisions between the LSS nodes and more general volume representations such as spheres, cubes or polygons, which can be used to model the rest of the environment. Thus, it is possible to model not only collisions and self-collisions between characters but

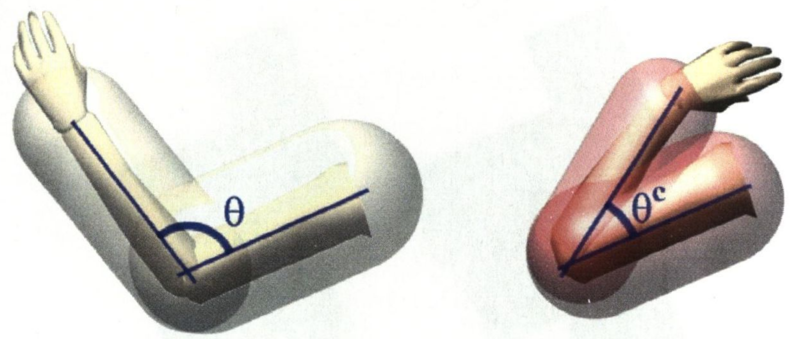


Figure 4.8: Self collisions between adjacent LSS nodes

also between the characters and a more generic virtual environment.

4.2 Compressing Contact Data

Approximate contact models deliver increasing numbers of *contact primitives*[1] which, when input into the collision response mechanism, result in a richer collision response model. An important part of contact modelling is returning data that the response model can handle in terms of processing time. In hierarchical collision detection, contact primitives tend to be more numerous at the finer levels of detail than in accurate collision detection. For instance, if we take the example of two planes in contact, this may be dealt with as a single contact primitive by an accurate collision detection algorithm, *i.e.* a face-face collision (see Figure 4.9). This is not terribly disturbing as it is certain that at some coarser levels of detail, refinable collision detection would also treat this as a single collision pair (two single bounding spheres).

It should be noted that BVH contact primitives are generally computed faster and therein lies the trade-off between BVH collision detection and exact collision detection methods. However, when the branching factor is significant it becomes

[1]We will define a *contact primitive* as the collision direction and collision point pair that we have already been using for response.

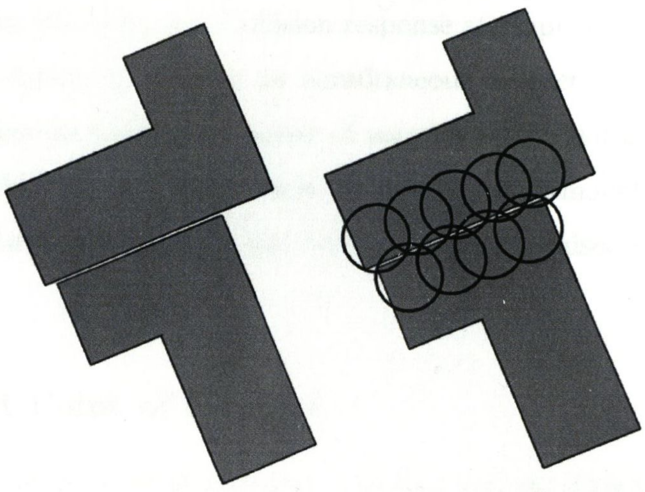


Figure 4.9: A single face-face collision may sometimes be detected as multiple BVH node collisions

prohibitive to calculate response on the large numbers of primitives that are output by the contact modeller. In such cases the bottleneck is in the response module, *i.e.* the contact modeller can cheaply compute larger numbers of contact primitives within the limits set by the scheduler but collision response is not quite able to handle as many. Thus, we need to reduce the contact primitives passed to the collision response module.

The simple solution would be to interrupt the contact modelling phase when we know we have gathered as much data as the collision response module will be able to handle. However, as the bottleneck is in the response model and contact modelling can in practice deliver much more than collision response can handle it would be desirable if we could somehow exploit the extra contact modelling data. Rather than limiting the number of contact primitives that contact modelling is allowed to compute, perhaps we could use the extra data and reduce it to deliver a smaller number of more accurate contact primitives to collision response.

Essentially what is required is the reduction of a larger set of contact primitives to a smaller one that is easier to handle but preserves the richness of the data that is represented by the larger set. Collision response attempts to solve the net force or impulse on objects by solving for simultaneous contacts. Normal reduction attempts to generate a reduced number of normals that when passed to collision response will achieve the same net result. Heuristic contact modelling methods can be used to perform this reduction and enable a more optimised collision response output.

4.2.1 First Point of Contact

In some cases where a BVH collision detection system detects multiple node penetrations, it is possible that some of these occurred only due to the over-approximate nature of the bounding volumes combined with the discrete nature of the time steps. For instance, Figure 4.10 shows an object moving in a straight line downwards with no rotational velocity. Although many nodes are interpenetrating at the instant where the snapshot has been taken (Figure 4.10), response should really have been applied at an earlier time. If we can assume that this is usually the case then a good strategy to contact point reduction is to seek out the earliest contact nodes in a multi-node collision event. A reasonable approximation as to which node collided first can be obtained by inspecting the interpenetration depths of each colliding node.

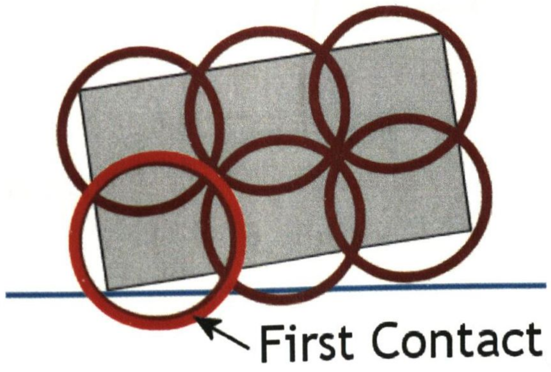


Figure 4.10: Two colliding nodes, deeper interpenetration is taken as first contact

In our new scheme, we require one added detail in the contact primitives, namely the interpenetration measure, K. The theoretical value of K should be obtained be calculating the volume of the interpenetration region (we will call this the *volume of intersection*) between the two volume nodes and comparing this with the volumes of the nodes themselves. As shown in Equation 4.1, we take K to be the ratio of the volume of intersection to the volumes of the smaller of the two nodes ($\boldsymbol{V_a}$ and $\boldsymbol{V_b}$ in Equation 4.1). This is made clear in Figure 4.11.

$$k = \frac{V_a \bigcap V_b}{min(V_a, V_b)} \tag{4.1}$$

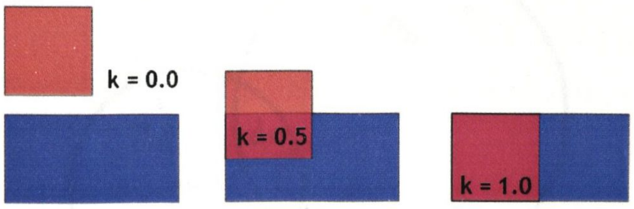


Figure 4.11: Interpenetration ratio

However, calculating the volume of intersection is an expensive process so ideally we would like a measure that relates more to a distance. If we can assume that most BVH nodes will be regular solids, whose volumes are roughly proportional to the cube of their radii, and extend this assumption also to the volume of intersection, which will have an effective radius d_K (the "diameter" of interpenetration) then we have:

$$k \propto \frac{(\frac{1}{2}d_K)^3}{(min(R_a, R_b))^3} \tag{4.2}$$

where (see Figure 4.12):

$$d_K = \frac{R_a + R_b - d}{2}$$

In practice we can remove the cubic powers in Equation 4.2 without any serious consequences, since all that we require is a scalar value that will reasonably allow us to quantitatively compare two different levels of interpenetration, and that converges correctly to the limits of 0 and 1 for "no interpenetration" and "full interpenetration" respectively. We then get a measure for interpenetration κ, that is considerably quicker to compute:

$$\kappa = \frac{d_K}{2(min(R_a, R_b))} \tag{4.3}$$

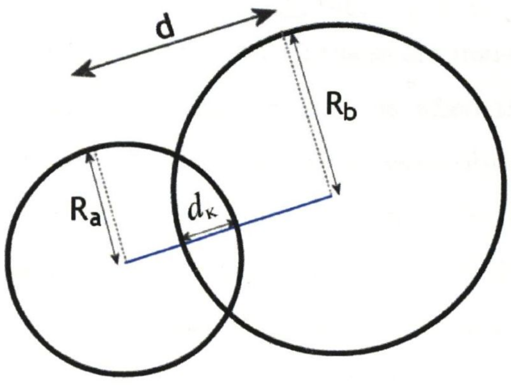


Figure 4.12: Calculation of diameter d_k

4.2.2 Simple Averaged Normals

For non-concave pairs of objects where many adjacent nodes of similar orientation are colliding (*e.g.* the nodes of two almost-parallel colliding faces), an effective solution is to average the normals returned by contact modelling as shown in Equations 4.4 and 4.5. Figure 4.13(a) shows two colliding BVH trees for which the N contact primitives have been calculated and averaged (Figures 4.13(b) and 4.13(c) respectively) according to the following simple equations:

$$\mathbf{p}_{ave} = \frac{\sum \mathbf{p}_i}{N} \tag{4.4}$$

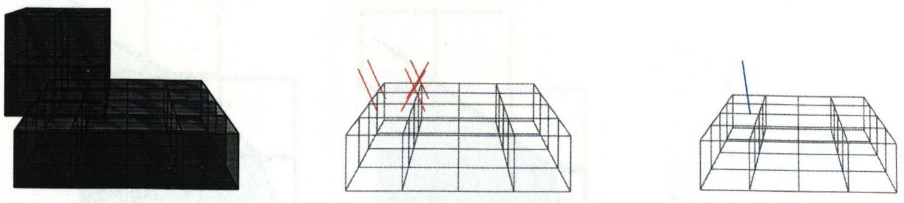

(a)BVH Trees (b)Multiple Contact Normals (c)Averaged Normals

Figure 4.13: Reducing collision data by simple averaging

$$\hat{\boldsymbol{n}}_{ave} = \frac{\sum \hat{\boldsymbol{n}}_i}{|\sum \hat{\boldsymbol{n}}_i|} \tag{4.5}$$

Simple averaging is adequate when the surfaces are non-concave and the multiple points of contact are close together, such as when they are derived from collisions involving adjacent volume nodes on the same object. In the example of the two faces, this is in fact a very accurate solution as the difference between the different normals is negligible and this will deliver a normal that is similar to the normal of the two (in this case close-to-parallel) object faces.

When the contact manifold is concave or if there are disjoint groups of colliding nodes, this approach has a high probability of return erroneous results. We therefore need to ensure that if normals are to be reduced in this manner that we identify the cases where it is safe to do so.

4.2.3 Pre-processed Grouping

An effective solution is to pre-group BVH nodes in the tree at the pre-processing stage. When the tree is initially generated, we store information in the node data-structures about which polygonal primitive they should be associated with. When we find, in the collision detection phase, that the collision involves nodes of the same group, we know immediately that it is safe to average them.

BVH generation is a topic worthy of study in its own right so we do not provide a detailed description here, but simply refer to previous work in the area:

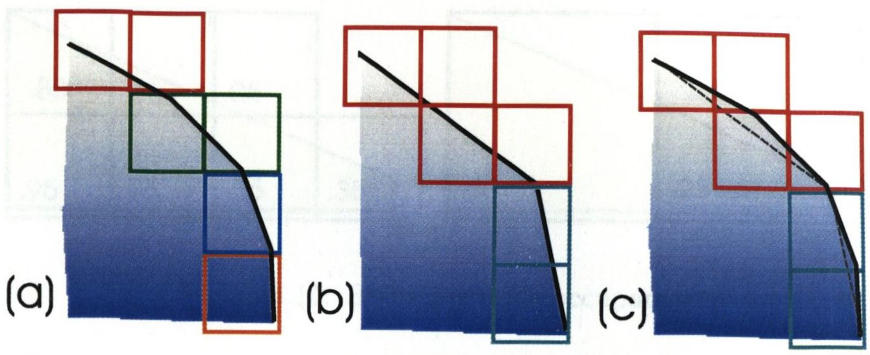


Figure 4.14: Assigning groupings based on a reduced mesh

[7, 8, 47, 48]. It is enough to point out that, as the BVH trees are generated from the original definition of the object, which in most cases will be a polygonal representation, it is simple to assign to each BVH node an index to the polygon that it bounds and use this for grouping volume nodes that lie on a plane as described in the previous section. Where the original model is not polygonal, *e.g.* an implicit/parametric surface or even a volumetric description, then groupings can still be obtained in the pre-processing stage by first generating a polygonised representation of the object and proceeding as before.

For high resolution meshes, it is more efficient to apply mesh reduction before grouping, as the approach described in Section 4.2.2 is feasible for almost-planar contact manifolds. Figure 4.14(a) shows grouping of surface nodes for a bounding hierarchy. Groupings are colour coded by group. Figure 4.14(b) shows groupings for a reduced mesh and 4.14(c) shows the reduced grouping mapped back on to the high-resolution mesh.

4.2.4 Calculating Normals from Density Gradients

Further optimisation is made possible when we consider the fact that volume nodes are inexact representations of the underlying object. Not only do the nodes approximate the underlying object but they do so inconsistently. In other words, we should consider a value called the *occupancy* (or density) of a volume node

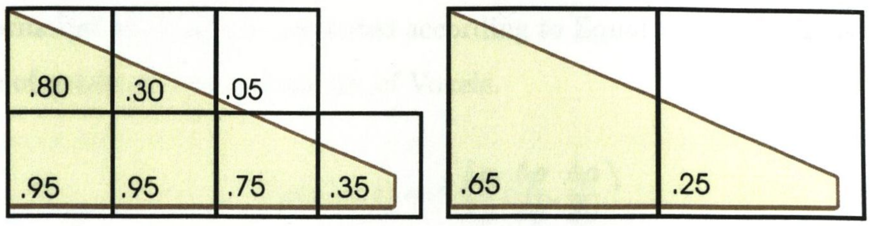


Figure 4.15: Bounding node occupancy

ρ_i, which is the percentage of the node volume that is actually occupied by the physical object that it is approximating. We will find that there can be significant variance in the values of $\rho_1, \rho_2, \rho_3, ...$ even though they might be sibling nodes in a homogeneous tree (see Figure 4.15). He and Kaufmann use a similar metric which they refer to as the surface crossing probability [40].

This highlights the concept that when an intersection is detected between volume nodes, this only signifies a *possible collision* between those two nodes. The probability that a volume intersection actually signifies a collision between the bounded objects is proportional to the occupancies of the two intersecting volume nodes (ρ_a and ρ_b respectively) and to the degree of interpenetration between the two nodes, κ.

$$P_{colliding} = \kappa \times \rho_a \times \rho_b \tag{4.6}$$

With the added detail of the interpenetration measure, this is in fact simply a generalisation of the bounding volume contact modelling described in Chapter 3, which always assumed $\rho = 1$ and $\kappa = 1$. Basically this means we have been assuming a 100% chance of intersection when any of the bounding volume nodes is found to intersect. We will see precisely how the probability value $P_{colliding}$ is used in section 4.2.5.

In volume graphics, it is a common approach to derive normals from voxels by calculating the density gradient for different voxel samples [58]. This normal is then used in a manner similar to the surface normal for rendering purposes.

The normal at any point is calculated according to Equation 4.7 by inspecting the change of gradient over a small set of Voxels.

$$\nabla\rho(x,y,z) = \left(\frac{\delta\rho}{\delta x}, \frac{\delta\rho}{\delta y}, \frac{\delta\rho}{\delta z}\right) \tag{4.7}$$

Using this in a descretised volumetric data structure, we have:

$$\nabla\rho(x_i, y_i, z_k) \approx \begin{bmatrix} \frac{1}{2}(\rho(x_{i+1}, y_j, z_k) - \rho(x_{i-1}, y_j, z_k)) \\ \frac{1}{2}(\rho(x_1, y_{j+1}, z_k) - \rho(x_1, y_{j-1}, z_k)) \\ \frac{1}{2}(\rho(x_1, y_j, z_{k+1}) - \rho(x_{i-1}, y_j, z_{k-1})) \end{bmatrix} \tag{4.8}$$

$$\hat{\boldsymbol{n}}(x,y,z) = \frac{\nabla\rho(x,y,z)}{|\nabla\rho(x,y,z)|} \tag{4.9}$$

A similar approach can be taken for volumetric collisions. The occupancy metric for each volumetric node is equivalent to the density of the volume node and a similar approach to that used in volumetric rendering can be used to calculate an approximate normal $\hat{\boldsymbol{n}}(x,y,z)$ at any point based on the occupancy gradient.

Although this approach can yield meaningful results for the normals, it is somewhat inconsistent with the requirements we stated in Section 3.2. Our collision response assumption was that the collision normals should be parallel for the two colliding objects (in opposite directions along the same directional vector). Calculating the normals using the density gradients unfortunately results in collision normals for each object being calculated independently of the object that it is in contact with. Such a problem does not arise in Volume Graphics because only a single object is considered at a time and basically it is a static normal that is required.

However, the concept of using a density metric in normal calculations is a sound one. We simply need to take into account that, for collisions, it is important not to consider just the individual objects but also the objects that they are colliding with. Also it is important that we consider the spatial properties of both of the individual colliding objects at the time of collision (*i.e.* their positions and orientations).

One possibility is to take the normal for each pair of colliding nodes to be the average of the two pre-computed density gradient normals of the two nodes:

$$\hat{\boldsymbol{n}}_A = \frac{\hat{\boldsymbol{n}}_b - \hat{\boldsymbol{n}}_a}{2} \tag{4.10}$$

where $\hat{\boldsymbol{n}}_A$ is the collision normal between nodes a and b; $\hat{\boldsymbol{n}}_a$ and $\hat{\boldsymbol{n}}_b$ are the normals at a and b precalculated from density gradients. Note that these will be pointing in opposite directions for two colliding objects which is why $\hat{\boldsymbol{n}}_a$ has a negative sign.

This is relatively quick over the single pair of colliding nodes, but for multiple collisions we still need to take the extra step of averaging over all the pairwise-averaged normals. The approach provides reasonably quick results with plausible normals that are adequate for non-interpenetration constraints when the evaluated BVH groups are simple and regular, as in voxel arrays similar to those used in volume rendering. Certain classes of sphere-trees *e.g.* regular Octree-based sphere-trees are also suitable. However, many classes of generic BVH trees do not impose constraints of uniformity on sibling spheres in the hierarchy [7, 47, 48, 37, 23]. Not only is this a problem because the volume elements are no longer discretised in such a way that we can easily step through a one-dimensional list of neighbouring nodes, but it would seem suspect to calculate a gradient from what is effectively an inconsistent step-size through this list of nodes.

4.2.5 Weighting by Interpenetration and Density

Another logical step is to use not just the node occupancy but the collision probability in Equation 4.6 as the factor for perturbing the collision normals obtained from direct normal approximation. We take into account how much individual nodes have interpenetrated as well as the node occupancy and use the probability of collision (see Equation 4.6) to get a weighted average of a set of pre-grouped normals:

$$\hat{\boldsymbol{n}}_A = \frac{\sum P_i \hat{\boldsymbol{n}}_i}{|\sum P_i \hat{\boldsymbol{n}}_i|} \tag{4.11}$$

$$\mathbf{p}_A = \frac{\sum P_i (\mathbf{p}_i - \mathbf{p}_{ave})}{\sum P_i |\mathbf{p}_i - \mathbf{p}_{ave}|} \tag{4.12}$$

4.3 Heterogeneous Contact Modelling

Now that we have the necessary procedures in place for contact reduction and for different kinds of contact, it is possible to set up a generic collision handler for the various different types of contact primitive families.

The higher level system remains the same, where collidable objects during a simulation are passed through broad-phase collision detection and a certain number of these are queued on a list of potential collision pairs.

We set up a table of intersection procedures and contact modelling procedures for each possible node pair that we wish to support, *e.g.* voxel-voxel, voxel-sphere or sphere-sphere.

The collision list is prioritised or bucket sorted into discrete priority partitions and refinable collision detection then proceeds to perform collision detection in scheduled order by inspecting each collision pair, retrieving the relevant collision/contact model and either eliminating the node from the list or refining it to a higher level of accuracy (see Figure 4.16).

4.4 Summary

In this chapter we discussed several aspects of the contact modelling problem, extending the framework described in Chapter 3 to efficiently handle more general cases of contact primitives. The key points are highlighted below:

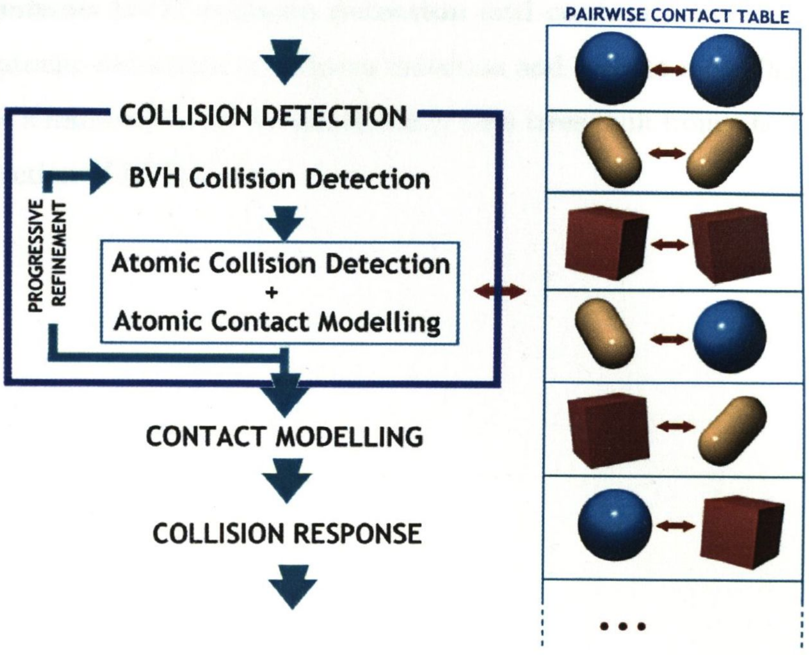


Figure 4.16: Framework for heterogeneous BVH collision detection

Strategies for contact primitive reduction : although the increased number of contact primitives sometimes returned by BVH collision detection is not harmful to the system in terms of deriving an accurate collision response, resolving multiple simultaneous contacts is an expensive task. We have provided heuristic optimisation strategies to reduce extra workload from collision response.

Alternate BVH schemes : we extend the capabilities of the BVH Collision Handling system in Chapter 3, which was originally based on a sphere-tree approach, to include BVH's of Sphere Swept Volumes and Voxels.

Contact modelling for articulated figures : using an SSV-tree together with an articulated transform hierarchy, it is possible to extend our system, originally designed to deal with rigid bodies, to detect and model collisions and self-collisions involving articulated figures.

Heterogeneous BVH collision detection and contact modelling : based on atomic definitions of collision detection and contact modelling we introduce a framework that works efficiently with trees built from a heterogeneous collection of BVH nodes.

Chapter 5

Collision Scheduling

Previous chapters discussed how target frame-rates can be guaranteed by using time-critical mechanisms for computationally expensive parts of the simulation process. Incremental mechanisms are used to generate results of increasing accuracy as more time is spent on processing. Although such systems may guarantee target frame-rates, they do not, on their own, ensure that the trade-off between accuracy and processing time is optimised. For this we must incorporate some form of prioritization within the process. Certain events, or specific parts of the scene, are categorised as being more important and given more processing time as a result. Prioritisation of the scene is based upon factors related to visibility and perceptibility of the approximations made to different parts of the scene. Only then is the speed-accuracy trade-off optimised by an interactive scheduler, which distributes the computational effort to different processes, ensuring that the most important parts of the scene are dealt with at the highest level of accuracy.

This chapter discusses how the perceptual priority of scene elements can be measured as well as techniques to use this information to optimise perceptual trade-off within an animated scene. We also discuss some problems associated with adaptive level of detail and strategies for minimizing simulation artifacts.

5.1 Collisions and Perception

Mirtich found that it took 97 seconds on average to compute each frame of an avalanche simulation of 250 convex objects on an SGI Onyx (200 MHz R10K CPU) [66], not because of any shortcomings in his simulation algorithm, but because of the high number and complexity of contact groups formed. His implementation favoured robustness over efficiency and for some applications this may be necessary. However, for real-time animations such a sacrifice is not an option and it is obvious that a trade-off between detection accuracy and speed is necessary to achieve a high and constant frame-rate. We now ask: what effect will this have on the viewer's perception of the resulting physics?

For scenes as complex as Mirtich's avalanche simulation, a great deal of simplification is required in order to even come close to achieving real-time frame-rates. Wherever trade-offs are required, it is desirable to achieve them in the most optimal way possible. In the Physically Based Animation domain our goals are as follows:

i. we need to guarantee target frame-rates for real-time performance regardless of scene complexity

ii. we must handle as much complexity as allowable within any given processing time limit *e.g.* a target frame-rate

iii. where simplifications become necessary, we should aim to get the best return in processing cycles for any details that are sacrificed

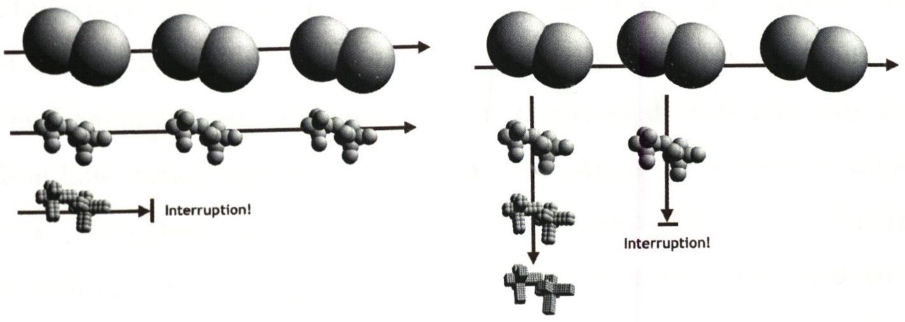
iv. we must minimise the overhead costs for the part of the system that manages the trade-offs between complexity and speed

5.1.1 Perceptual Optimisation

The first requirement for a system that will trade-off accuracy and speed is a mechanism that can handle processing at different resolutions and return consistent results. Such a system, as discussed in Chapter 3, is now available to us and allows us modulate the degree of accuracy in a dynamic scene but we have yet to define where these simplifications should take place within the scene.

The time-critical mechanism for collision detection, for instance, will allow us to guarantee a given target frame-rate by interrupting the refinable process of contact modelling when the time-quota has been expended, but we should consider two different ways in which this could happen:

Firstly, if we were to simply consider collisions in random order and process colliding trees in a depth first manner (Figure 5.1, then when the time-critical process is interrupted we would find that a few collisions are processed at very high levels while many more are processed at a very basic level or not at all due to the fact that so much time was spent on processing the first few collisions. Alternatively, a fairer approach would be to process colliding nodes in a breadth first manner, so that all collisions receive fairly equal treatment until the scheduler again halts the process.



(a) Breadth-first (b) Depth-first

Figure 5.1: Traversal Strategies

However, in large-scale simulations this is often not the most fruitful approach.

Consider, for example, a scene where many objects are squeezed into a small space leading to large numbers of collisions. In such a case, it is likely that if processing time were distributed equally among all collisions then, because of the large number of events that need to be considered, they would all most likely be dealt with at a very coarse level of detail. Although this might seem like the logical approach to take from a general viewpoint, this is not the optimal solution when we consider the perceptual significance of the simplifications we make. When we know that simplification can not be avoided, then our primary goal shifts from trying to achieve full mathematical accuracy to delivering the perceptually most plausible output that is possible in the given time constraints.

In a complex scene with many different interacting objects, it is often the case that certain objects or certain regions within the scene can prove to be more important than other regions and ideally should be given more detail at the cost of conversely less important objects or regions, which we can handle at lower levels of detail in order to save processing cycles. The solution we advocate is to prioritise events across the scene and distribute processing time based on this prioritisation.

5.1.2 Prioritisation Metrics

When there is adaptive trade-off within a system, the level of optimisation depends a great deal on the ability of the system to quantify the good or harmful effects resulting from any chosen trade-off. In other words, what we require are accurate models of how certain factors influence user perceptions of the scene or objects in the scene. *Perceptual metrics* are often used to evaluate the quality of perceived output such as from a computer graphics rendering or animation and are often used as bases for comparing outputs (*e.g.* renderings or animations) resulting from different approaches. With reference to adaptive levels of detail, reliable metrics can also be useful in deciding which parts of the scene are perceptually less significant and thus better candidates for simplification.

In interactive animation, meaningful results have already been achieved by using simple well known metrics such as object velocities, projected screen distances and distance from the user's fixation point (determined with the use of an interactive eye-tracker) [30]. More extensive studies need to be performed, however, to identify the most important factors which affect user perception of simulated dynamical events and to determine how several different factors can together influence the perceptual quality across a simulation scene.

There is some literature on the subject of perceptual analysis of graphical scenes but, to date, the majority of published work deals with the perception of static rendered images. In animation there have been studies that deal with levels of detail associated with the real-time rendering of the scene or elements within the scene [30][81] rather than the perception of motions synthesised by a dynamics system. There have, however, been several studies of how people perceive dynamical events in a more general context and some of the results should apply to perception of dynamics in a virtual world. As discussed in Section 2.3.1, there is evidence to show that humans from very early on in life incorporate dynamics into their perceptual model of the world [57] but also that humans are generally incapable of accurately detecting anomalous physical behaviour when there is more than one dimension of information involved [34].

Much of this literature deals with general dynamics perception but there is some previous work which deals specifically with the perception of dynamics in a virtual world. Barzel *et al.* [6] suggest exploiting uncertainty in order to save processing cycles in a physically based animation system. Reddy discusses certain areas where trade-offs are made possible in general animation [81] while Kaiser [49] and O'Sullivan [74, 75] specifically discuss dynamic scenes. O'Sullivan [74] describes a study of how effective users are at detecting gap distances in planar collisions between two objects at different speeds, angles of eccentricity and in the presence of different numbers of distractors in the scene. In [75] this model of collision perception is used in an animation system which trades-off collision accuracy with speed. However, these initial studies dealt with objects moving in a 2D plane orthogonal to the viewport and responding to collisions using a very much simplifed model of physics. As a result, they did not incorporate the effect of perturbations due to simplification in the actual physical behaviours of the colliding objects.

Similar studies would be useful, in a 3D dynamic virtual environment, where it becomes necessary to trade-off accuracy in the simulation processes that take place in the system. In Section 6.2 and the related paper [73] we discuss a set of experiments to understand user sensitivity to anomalies in dynamic simulations and provide useful data in order to evaluate the effectiveness of the trade-off on the believability of the scene. In order to decide in which parts of the environment we should apply trade-offs, it is important that we have fast metrics that quantify how a specific improvement or simplification affects the perceptual quality of the simulation. For example, O'Sullivan [75] found that erroneous collisions in the periphery of a viewer's point of fixation are less likely to be detected, while collision anomalies that occur in the presence of increasing numbers of visually homogeneous distractors are also less noticeable.

5.2 Metrics for Adaptive Simulation LOD

A good study of some common factors that can influence user perception of individual scene elements in an interactive animation is provided by Reddy [81]:

Distance : the object space distance between the viewer and a scene element is easily computed and is a useful measure of relative perceptual importance. This is a widely used metric for adaptive simplification *e.g.* in choosing terrain triangulation in outdoor simulations

Size : viewer to object distance alone is sometimes inappropriate for prioritising scene elements when they are not of comparable dimensions. A more suitable metric is the projected size of the scene element.

Eccentricity : this is described by Reddy as "the degree to which an object exists in the visual periphery". If the assumption can be made that the viewer will most likely be looking at the centre of the screen then this is simply the projected distance of the scene element from the centre. The use of eyetracking equipment or gaze-prediction algorithms makes it possible to more exactly define the user's fixation point and better measure eccentricity.

Velocity : in animation scenes the speed of motions can affect the viewer's ability to notice certain simplifications. Once again, a more meaningful measure is projected velocity.

If we can give each collision some measure of importance, the next logical step would be to sort all collisions and apply increasing levels of refinement to each in order. However, it is usually the case that the overhead of a full sorting process is computationally very expensive, thus reducing the gains made from not using traditional exact collision detection techniques [75, 74]. A more fruitful approach is to partition the set of collisions into discrete subsets based on the prioritisation criteria. A similar approach was taken by Duchaineau [27], who maintained priority queues to select which regions of a mesh to simplify while rendering terrain in real-time. In our system, collision processing is first applied at a low attainable level to all collisions and then collisions in the higher priority subsets are refined to whatever level is possible in the allocated time remaining. In our implementation, this was achieved by storing collision data in two separate priority lists. All collision events are represented as collision data structures in either one of the two lists based on some importance criteria. Collisions in the high-priority list are allocated more processing time so that the contact model and resulting response is more believable.

5.2.1 Application of LOD metrics to Collision Handling

This section discusses the application of these factors to Adaptive Collision Handling. As we are dealing with collisions, we should note that what is being sorted in perceptually adaptive collision handling is actually collision pairs rather than the objects themselves. So the value returned by the prioritiser needs to be a function of both objects in the collision pair.

Given any prioritisation strategy, we would ideally wish to sort all objects in the scene based on their priority and apply simulation refinement to objects in order of their priority values. However, in practice the computational cost of performing a complete sort can become unjustifiably prohibitive so a more practical approach is to use a small number of priority groups into which colliding objects are bin sorted with respect to certain threshold values. Each priority group is then allocated its share of processing time by the scheduler, with more processing being expended on higher priority groups. This method, whilst still preserving some level of prioritisation, bears considerably less overhead expense than a full continuous sort and in practice delivers improvements even with only two priority groups. We should beware of using too many variables, as the calculation of the priorities becomes a more expensive task, thus leaving even less time available for collision processing, as shown in [74]. We now discuss several candidate metrics and we will compare their effectiveness in Chapter 6.

Distance

Distance is perhaps one of the most commonly used metrics in LOD and also the most straightforward to implement. The Euclidian distance between the viewer and the object is simply taken as a measure of priority. For a fully sorted prioritisation scheme this is very easy to implement. However, with discrete priority regions we must answer the question of how to split our scene into the different regions so that the trade-off is best organised. Unfortunately, this is a very subjective issue and an ideal partitioning is dependent on various different factors such as the number of partitions, the viewing parameters, visibility in the scene and the number and properties of objects in the scene. As a result, it is fairly common to choose partitioning in an *ad hoc* manner or by statistical trial and error.

Distance as a metric has advantages in that the scene ends up being partitioned into spatial regions in object space. Therefore, objects in close proximity to each other tend to have the same allocated priority which can be useful in reducing some popping errors in the simulation. For a more detailed description of popping see Section 5.3.1.

The distance metric returns a value based on the Euclidian distance between the viewer and the centre of collision between two objects. The centre of collision is a term introduced by O'Sullivan [75] which, in our system, can be used interchangeably with the point of collision. For a collision pair, this is taken as the point of collision in the bounding sphere collision check.

Size

Size prioritisation is based on the projected screen size of the objects and therefore incorporates distance if we are dealing with perspective projection. Calculating a proper projected screen size of objects can be slightly more expensive than the simple Euclidian distance metric, but it is usually sufficient to preassign a one dimensional width in object space to an object and simply inspect how this projects on to the screen. A good value to take in our system is the radius of the coarsest bounding volume in the BVH tree. Unless objects in the scene are all of the similar object space dimensions, this invariably leads to priority regions not being spatially separated *i.e.* a large object further away might have a much higher priority than a small object up close while two objects close together might be assigned altogether different priorities if they are of different sizes. This makes the approach marginally more prone to popping but size as a metric is generally a better representation of the object's perceptual importance than distance.

Size as a metric for collision handling can be used by taking an average of the projected radii of the two objects in a collision pair. For collisions however, it is more meaningful to take the projected maximum gap size between the two objects as an alternative to traditional size-based LOD. As with the distance metric, a convenient method is to measure maximum gap size only for the top level of the bounding hierarchy. This is considerably less accurate than calculating the maximum distance between each potentially colliding node but saves on a critically large number of wasted distance calculations and, as the collision list is only sorted once before the refinable collision detection phase begins, the top level maximal gap size is the meaningful measure to take. As an alternative, in a fully sorted breadth-first approach, we could re-sort the collision list after each iteration of refinement, however we do not presently support this more expensive procedure.

Velocity

Velocity based prioritisation is also fairly easy to compute as our dynamics objects already have a velocity state variable that we can query. However, as with size, it is important to consider not the object-space velocity but the projected image-space velocity of objects. This requires taking into account the projection of the velocity vector of an object as well as any movements of the camera due to panning or rotating. Even more so than with the size-priorities, this leads to different object priorities being scattered randomly throughout the scene. The velocity metric returns a value based on the screen velocity of an object which is computed according as follows:

$$\begin{aligned} \mathbf{v}_{img} &= \boldsymbol{viewing\ and\ projection\ applied\ to}\,\mathbf{v} \\ \mathbf{v}_{view} &= \mathbf{v}_{img} + \mathbf{v}_{cam} + (\boldsymbol{\omega}_{cam} \times r_{cam}) \end{aligned} \tag{5.1}$$

where:

- $\mathbf{v}$: velocity vector of object in world space

- $\mathbf{v}_{img}$: projected screen velocity of object in static frame
- $\mathbf{v}_{cam}$: panning speed of camera
- $\mathbf{v}_{view}$: projected screen velocity of object
- $\boldsymbol{\omega}_{cam}$: rotational velocity of camera (about view direction)
- $\mathbf{r}_{cam}$: displacement of object from view centre

Since this is an image space measure of velocity, it is marginally more feasible to seek less subjective threshold values for partitioning scenes into different priority groups. However, there are still a number of variables that factor into the effect of velocity on perception such as the object's size, visibility and eccentricity. Reddy presents some studies on threshold velocity values for user perception of simple 2D stimuli of constant and spatial frequency moving down the screen. Future work is planned on determining a model for velocity based partitioning of a 3D dynamic scene. This would be required for the ideal perceptually-optimised system. For now we choose, once more, a subjective (*ad hoc*) partitioning of the scene in order to evaluate the computational competence of the system to deal with velocity based partitioning.

Once again, for collisions we are interested in a function of the two colliding objects under inspection so we take the relative speeds of the colliding objects and apply Equation 5.1 to get the camera dependent relative velocity of the collision pair.

Velocity is also implicitly a factor in determining the level of refinement of scene objects through the recursion capping mechanism discussed later in Section 5.3.2, which is required to preserve the robustness of the system. Recursion capping prevents the collision handling system from using fine detail nodes which might return erroneous results due to an object's high velocity, thus causing excessive interpenetrations. At high velocities, therefore, it is not only perceptually better to simplify the detail level of the scene elements but it is also required for robustness.

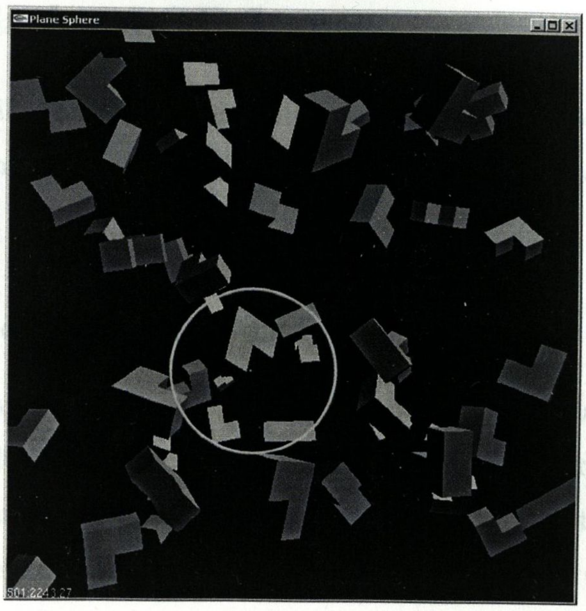


Figure 5.2: Collisions inside the region of interest receive more processing time.

Eccentricity

Eccentricity refers to the degree to which an object exists in the user's periphery. In a gaze-contingent prioritisation scheme, we can use an eye-tracker to determine the user's gaze location at any instant during the simulation. Eccentricity can then be used as the primary measure of importance, or combined with other factors as part of a more complete perceptual model.

Figure 5.2 shows an implementation of a system that uses eccentricity as the scheduling function. One high-priority list stores collisions that project inside a region of interest on the screen centered around the users fixation point (determined with an interactive eye-tracker), while less important collisions are relegated to a low priority list. The size of this region of interest may be adapted according to the number of visually homogeneous distractors present, *i.e.* the more distractors, the smaller the region. Alternatively, if eye-tracking is not feasible it is possible to use gaze-predicting algorithms to approximate the user's point of fixation or to take the centre of the screen as the focus point.

5.3 Simulation LOD Issues

This section describes a few common problems associated with adaptive simulation and BVH collision detection algorithms and discusses some possible solutions to the problems.

5.3.1 Level of Detail Popping

Level of detail in motion synthesis has many parallels with similar well-established techniques in multi-resolution and adaptive rendering. Unfortunately, some of the problems from traditional LODs are also issues in animation LOD. One of these is the problem of popping. Popping is the term in adaptive LOD rendering for the temporal artifact that occurs when a sudden change occurs between one level of detail and the other. Although both levels of detail might be acceptable representations of the object, they may contain slight differences and it is the change between the two models that often attracts the user's attention rather than the level of simplification itself. Popping is particularly noticeable when an object straddles the threshold between two regions of varying priority.

In animation level of detail and specifically in collision detection an example would be as follows:

- Two objects treated at resolution R move in parallel directions and almost touch.
- Due to a change in the scene their levels of detail both drop to $R-1$ and a coarser representation of the object's bounding volumes are taken.
- Objects are now found to be colliding because the coarse bounding volumes are intersecting and the objects repulse each other even though no change has occurred in their states relative to each other.

Different techniques lead to different degrees of popping. This is true in motion synthesis as well as in interactive rendering, so it is useful to have an idea of how suitable a certain technique is in relation to another for simplification or for prioritisation of scene elements. Common techniques to alleviate popping in Level of Detail include morphing or temporal blending or simply using levels of detail that are closer together and less distinguishable.

In interruptible simulation, although it is feasible to choose larger numbers of discretisations at different detail levels, it is not implicitly guaranteed that the chosen LOD for any scene element will not drop several levels of detail between one frame and the next. Temporal blending of a form is feasible, for instance, by using frame to frame coherency to preserve a object's priority level between frames where it straddles priority threshold boundaries.

Fortunately, BVH collision detection has some advantages, in this respect, over traditional LOD in that popping most frequently occurs when level of detail drops for an object. This is because an increase in detail, in most cases, leads to a more plausible and less noticeable result. This is a useful asymmetry because drops in LOD usually relate to objects leaving a region or a state where they are more likely to be noticed by the user. Furthermore, popping in collision LOD only becomes an issue for objects of they happen to be colliding at the instant that an LOD shift occurs and the user is observing the event. Thus the problem, in interruptible collision detection, is self-solving to degree but nevertheless, noticeable popping does occur and we should be mindful of simplification strategies that are more prone to it and take necessary measures to guard against it, such as frame-to-frame persistence of LOD until it is safe to allow a drop in resolution.

5.3.2 Tunnelling

Tunnelling is a term given to a common simulation error that occurs due to the fixed time-step weakness. Due to discrete solutions of the motions of objects, we may find that at high velocities object interpenetrations can occur to a degree that leads to lack of robustness in the simulation. Indeed, objects may sometimes pass through each other completely without the simulator ever picking it up. Figure 5.3 shows an example of tunnelling in a sphere-plane collision, the lowermost node of the sphere-tree has tunnelled through and approximate contact modelling has returned a normal for collision response that would drive the object further into the plane.

The problem can sometimes be avoided by ensuring that simulation time steps are small enough so that the degree of interpenetration is small and as a result we will detect it before it becomes critically detrimental to the robustness of the system. But the right time-step to choose depends on both the dimensions of the moving object and its velocity. For a generic system, however, we do not desire to have to place limits on either velocity or size. As for the time-step, this cannot unfortunately be made arbitrarily small. Furthermore, the problem is exacerbated by having BVH trees where the nodes of the tree, as a rule, become progressively smaller and therefore more prone to tunnelling.

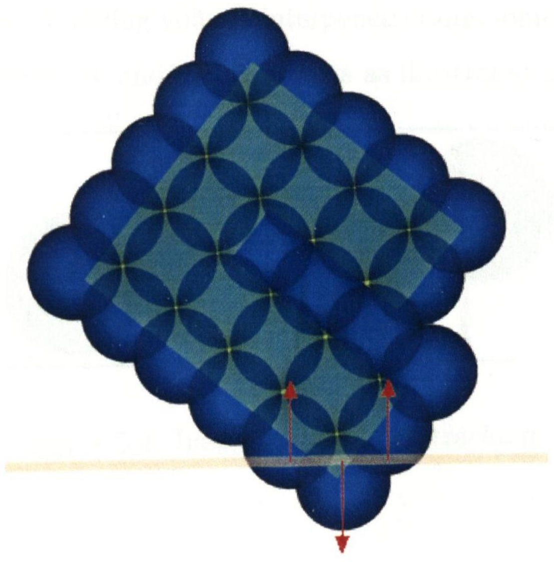


Figure 5.3: Tunneling of BVH nodes: calculated normal vector is in the wrong direction

Back-tracking

In some sense back-tracking in simulation time (retroactive detection) will solve the robustness problem to a degree. However, the tunnelling effect can still cause unexpected extra processing in some cases. For instance, if we were to back-track in simulation time due to object interpenetrations, there may be cases where this might reveal extra collisions/interpenetrations that were initially undetected due to tunnelling, thus requiring further back-tracking. Unfortunately, retroactive detection is difficult to incorporate into a refinable collision detection scheme. The reason for this is that back-tracking in simulation time basically involves discarding (when an interpenetration is found) all collision and contact data and restarting the collision process at a previous time frame where it is believed that the first contact has occurred. Furthermore, computing a good guess for the time of first contact is difficult with the approximate model of contacts and with adaptive levels of detail. Stepping back based on approximate collision data is also illogical because bounding volume interpenetrations sometimes do not imply an actual collision with the underlying objects as illustrated in Figure 5.4.

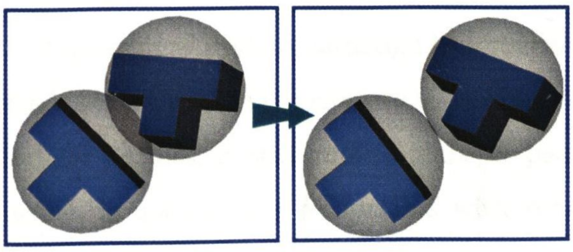


Figure 5.4: Inappropriate Back-tracking

Conservative Advancement

Mirtich [66] gives the name Conservative Advancement (CA) to a class of simulation techniques which, in some form, attempt to implement adaptive time-steps in order to reduce errors resulting from discretised simulation time. The strategy behind this is to predict the likelihood of simulation discontinuities (such as object collisions) occurring in upcoming simulation frames due the current state of objects in the scene. If the system perceives no such imminent discontinuities then it continues constantly with perhaps a relatively large time-step. When a possible event is expected, the system then attempts to adapt the time step to a safe level, *e.g.* halve the time-step, so that around the instant where the discontinuity occurs, simulation is performed at a much higher level. The system might progressively shorten the time-step (*e.g.* repeatedly halve the time-step duration) thus "creeping up" to the discontinuity as described by Mirtich. The goal behind all of this, from a collision detection perspective, is to try to get below the threshold of acceptance ε the very first time the collision is detected without requiring complicated back-tracking.

The predictive part of this is usually achieved, if not by analytical methods, by using intersection tests between coarse space-time bounds of the simulation objects. Space-time bounds are discussed later in this section.

Conservative Advancement and Interruptible Collision Detection

Referring back to the time-critical system, although it is conceivable to have time-critical conservative advancement (CA) (*i.e.* cease the process of time-step refinement when the scheduler decides, it still implies multiple passes of the collision detection system, which is not entirely compatible with refinable approximate collision detection.

We can, however, use a method that borrows from this type of technique in order to reduce BVH node tunnelling. This is done by putting a cap on the recursion through a BVH tree based on the likelihood of tunnelling occurring due to a number of factors, including the current time-step, the object's linear $\mathbf{v}_{node}$ and angular velocity ω, the position of the node r_{node} in relation to the centre of mass of the object it is bounding and the size of bounding volume nodes at the current level of refinement R_{node} (see Algorithm 5.3.1). Although this is different from normal CA approaches, it takes into account the same problems that CA hopes to avoid and effectively incorporates adaptability in collision detail in order to address the problem. This is illustrated in Figure 5.5.

Algorithm 5.3.1: CAPRECURSION(*node*)

```
$\mathbf{v}_{node} = \mathbf{v} + (\boldsymbol{\omega} \times \mathbf{r}_{node})$
if ($R_{node} <$ |timestep $\times\ \mathbf{v}_{node}$|)
  then stop refining for this object
```

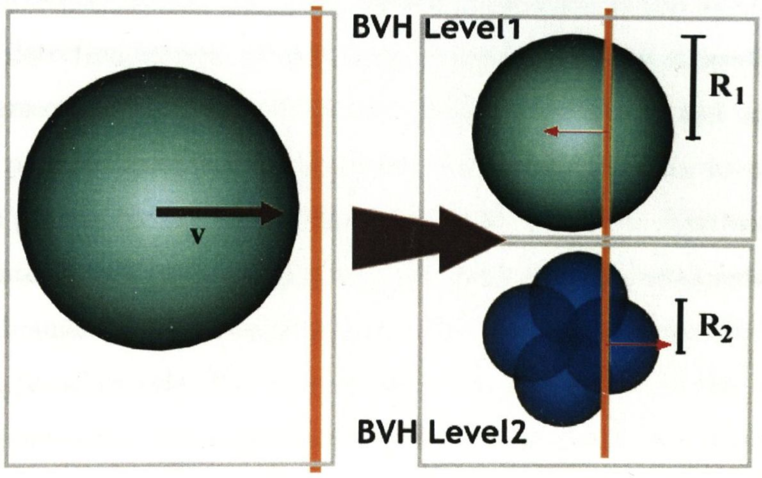


Figure 5.5: Recursion capping is required when an object's velocity is large in relation to the node radius (level 2 spheres)

Space Time Bounds

Perhaps the only sure way to completely eliminate tunnelling is to solve the problem in a non-discrete manner. This could be done by analytical collision detection, *i.e.* mathematically solving for points of contact and collisions using the mathematical descriptions of an object's temporal spatial occupancy. This is, of course, an expensive process and not presently workable in real-time for any significant number of objects. An alternative to this is to use coarse space-time bounds to detect when objects may have intersected. Space time bounds were discussed briefly in Section 2.2.1 we discuss them further here.

Space-time bounds are representations of the spatial occupancy of an object over time. Similar to bounding volumes, they are usually over-approximations of all possible spatial regions that an object might occupy in a given time interval. If, given an upper bound on acceleration, two objects are going to collide within a certain time interval, it is guaranteed that their space time bounds must intersect within this interval. Thus, if it is possible to perform quick intersection tests between space-time bounds of objects, it becomes possible to cull much processing on collision detection between objects those two objects if their space-time bounds are not intersecting. Although the concept is four dimensional and not trivial to solve, it is possible to optimise the process sufficiently in order to use it as an intersection process for the broad phase of collision detection. Hubbard discusses the use of intersection tests between parabolic horns and hypertrapezoids used as space time bounds in order to deliver speed-ups in collision detection [44, 47].

Some approaches take 3D volumes swept by the motion of the object as a coarser representation of the space time bounds of an object [66]. The space-time bound intersection test becomes easier to conceptualise and we can use known 3D intersection tests. This usually involves much coarser over-approximations of the object's positions as a 3D volume with cheap intersection tests, *e.g.* spheres and cubes, but is sufficiently useful for culling collision detection computation and detecting possible cases of tunnelling.

A coarse space-time bound is generated by taking a sphere with its centre at the object's current position and its radius us the upper bound on all possible changes in position due to the object's state and volume. The radius of a bounding sphere for the object R_ω is calculated. This is centered at the object's centre of gravity and bounds the object in all possible orientations. Then, an upper bound on the possible displacements of the object in a single simulation timestep is calculated based on the object velocity and forces currently acting upon it,

$s_Y(\Delta t)$. The radius of the spherical space time bound R_s is then obtained based on the dynamic state of the object Y as in the following Equation 5.2.

$$\begin{aligned} s_Y(\Delta t) &= |Y.acceleration| + |Y.velocity| \\ R_s(\Delta t) &= s_Y(\Delta t) + R_\omega \end{aligned} \tag{5.2}$$

Although this is an excessively coarse representation, it useful for detecting cases of complete tunnelling, *i.e.* an object at very high velocity completely passing through an object and not being detected for collisions even when it has emerged on the other side.

5.4 Summary

In this chapter we discussed strategies for scheduling adaptive collision handling in order to get an optimised trade-off between speed and accuracy. This is done by taking into account perceptual factors that influence a user's experience of the scene. This is a logical approach to take, because it is the output from the system and, more importantly, the user's experience of it with which we are most concerned. Thus, using even a coarse understanding of the factors that influence user perception as input into the scheduler leads us closer to optimising the speed-accuracy trade-off. We have explored four basic factors that affect perception of an animated scene and offered approaches to using them as input for the interactive scheduler. Although these have been previously been discussed as metrics for scheduling interactive rendering, we use them in the specific context of adaptive collision handling. In Chapter 6 we present some comparisons and evaluations relating to these.

We also discussed some problems unique to adaptive collision handling and solutions, which are often relevant to the scheduling strategy used in modulating adaptive levels of detail.

Chapter 6

Results

This section reviews the issues that we have discussed in previous chapters, presenting some practical output and evaluations of the approaches undertaken. Evaluating the quality of graphical output is difficult to do objectively. Unless we have established methods of quantifying visual correctness it is nearly impossible to convey in figures how successful an approach has been. This is even more so in the case of computer animation, where the temporal features rather than static details are what we seek to model. Thus, it is important to have a perceptual basis for measuring plausibility in dynamic simulations.

In Section 6.1 we first illustrate how our system achieves different levels of detail based on the approaches outlined in Chapters 3 and 4. Section 6.2 details results of some experiments to measure perception of collision handling. Section 6.3 is a study of the scheduling strategies described in Chapter 5 and we also present experimental results from an evaluation of an eccentricity based scheduler which uses an eye-tracker to determine user fixation. Finally in Section 6.4 we show output from typical scenes with which exact collision detection methods generally have difficulty.

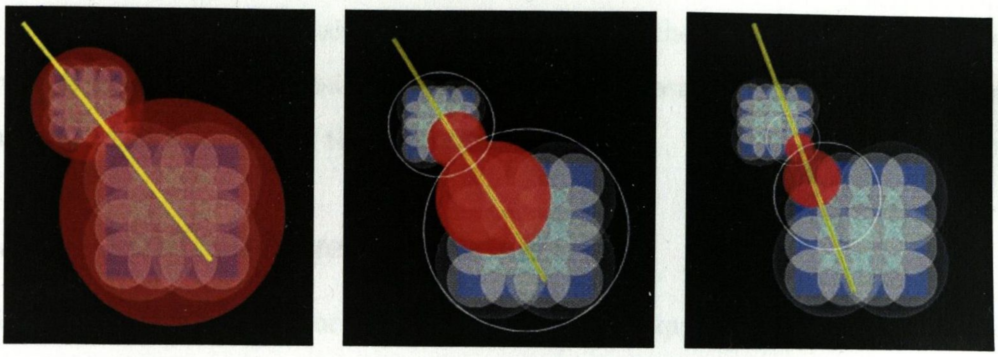


Figure 6.1: Contact Refinement in Sphere-tree

6.1 Levels of Detail

This section examines output from a refinable collision detection system and shows how accuracy of the collision handling data is gracefully degraded for a real-time system. We look at output from the contact modelling approaches discussed in Chapter 4.

6.1.1 Sphere-tree Levels of Detail

Figure 6.1 shows how increasingly fine resolutions of sphere-tree level of detail used in collision detection lead to a more accurate contact vector (shown in yellow). As a finer representation of the colliding objects is used for collision queries, we obtain a better approximation of the contact position and the direction of forces. The accuracy in modelling the contact vector directly affects the accuracy of collision response.

Figures 6.2(a)-(c) show sequences of frames from an animation of a single collision interrupted at three levels of sphere-tree detail. We look at the two dimensional projection of a very simple collision case for ease of visualization. The first two frames in each strip show, respectively, the starting positions of the two objects and the instant at which an intersection between sphere nodes is detected. After the collision has been processed, the three different animation levels of detail show increasing accuracy as can be seen from the trajectories and the rotational velocities of the two objects.

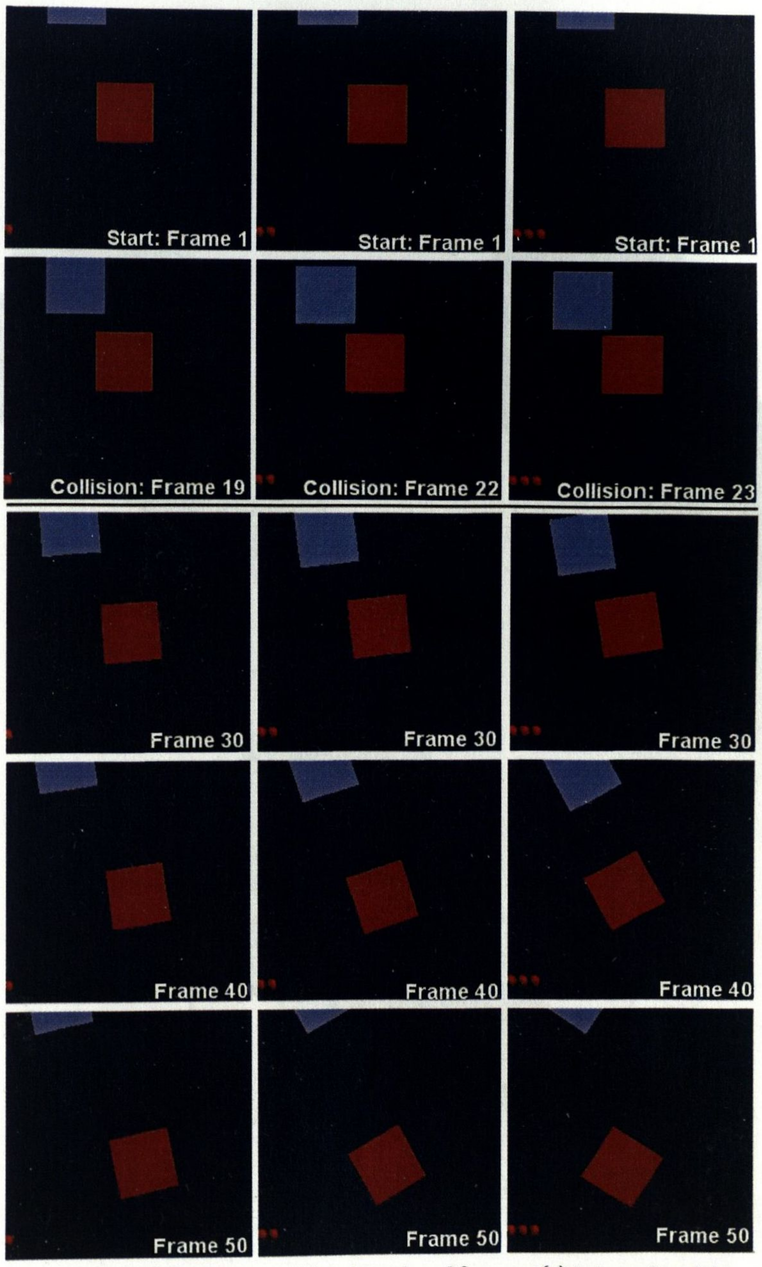
6.1.2 VoxTree Levels of Detail

Figure 6.3 shows contact normals resulting from the equations discussed in Section 4.2. In each figure, the red lines show the multiple contact primitive groups, while the yellow line is the single reduced impulse vector generated. In each case, the voxels have been given a density value based on the underlying mesh (outlined in white). Thus the larger flatter voxel block represents a prism shaped object. The cubic object is given a 100% density for all its voxels.

Figure 6.3(a) shows reduction by simple averaging over the voxel group. Note that for clarity we chose to show only the normals going from the prism shaped object outwards. Simple averaging does not in fact take the density of the prism Voxels into account and the contact primitives are simply averaged out to get a quick approximation of the reduced contact normal. Figure 6.3(b) shows the normals generated by the gradient averaging technique. Normals are first pre-computed for all the objects based on density gradient. For each two nodes that are found to collide, an average of the two normals is taken. These must then be averaged out to get the single reduced normal. In Figure 6.3(c), a weighted average is calculated based on the original (red) collision primitives to get the yellow reduced normal. Weighting is based on both the density of the colliding nodes as well as their interpenetration values and is represented in the image by the length of the red lines.

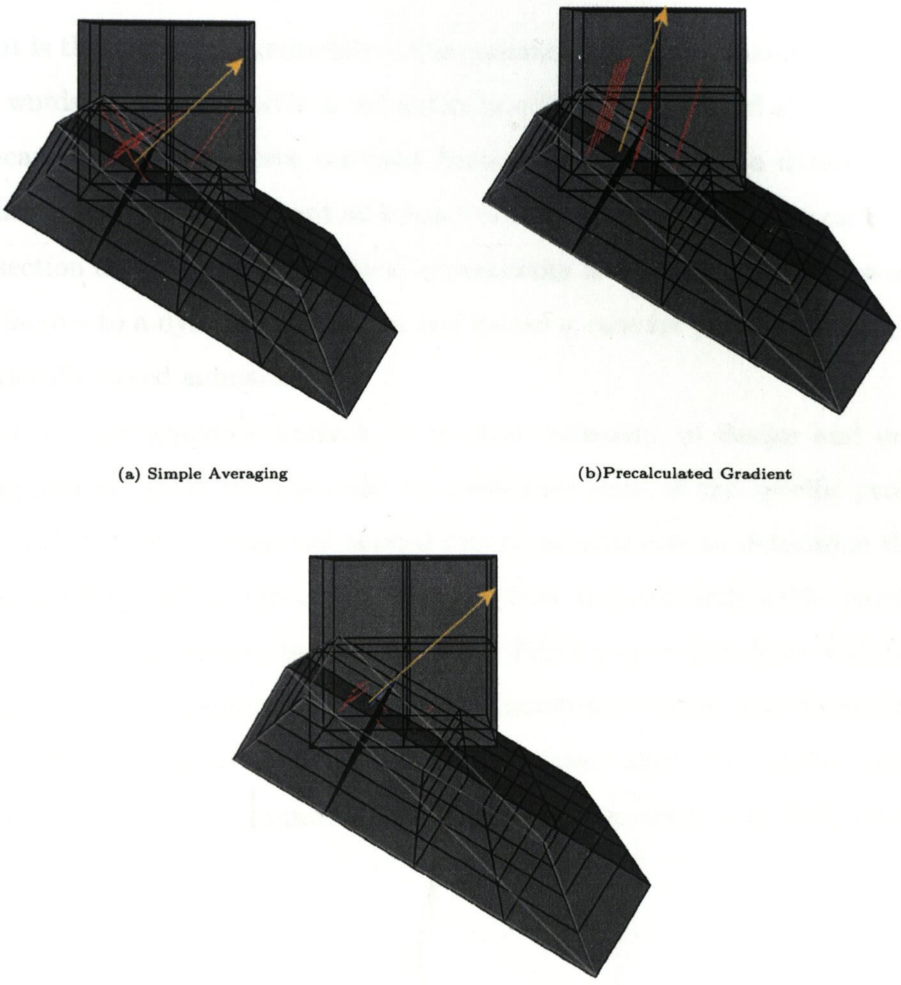
6.2 Psychophysical Experiments

Previous chapters advocated a system that trades off accuracy for speed in collision handling. The precise area of trade-off is in three main areas namely:



(a) Sphere Level 1 **(b) Sphere Level 2** **(c) Sphere Level 3**

Figure 6.2: Animation strips: Shown above are a selected frames from simulations at three different levels of sphere-tree detail. Note the varying gap at "collision time" and the differences in the calculated final linear and rotational velocities.



(a) Simple Averaging

(b)Precalculated Gradient

(c)Weighted Average

Figure 6.3: Contact Models with Vox Trees

- collision detection accuracy
- contact modelling/collision response accuracy
- framerate or more importantly frame-rate consistency

But is this trade-off worthwhile in the context of the full dynamics system? In other words, can we live with a reduction in collision detail and is it really very significant to try to preserve constant frame-rates? In order to determine this, we must investigate the perceptual impact of these individual factors on the user. This section discusses psychophysical experiments to investigate the relevance of these factors to a dynamic simulation and makes a case for perceptual adaptivity in physically based animation.

In computer graphics research, it is often necessary to design and execute psychophysical experiments in order to investigate some of the specific problems raised: [41, 92]. We carried out several sets of experiments to determine the extent to which the effects reported in the literature are applicable to the particular collision handling scenarios being considered. Previous research dealt with factors such as eccentricity, separation, presence and number of similar and dissimilar distractors [74]. We describe, in the following sections, additional studies designed to examine the effects of causality, kinematics and dynamics on participants' perception of collisions.

6.2.1 Causality

Sudden drops in frame-rate during collision events can have serious effects on the perception of an animation. As Michotte [64] demonstrated in his experiments, objects moving apart after a certain delay are no longer perceived to be doing so as a result of the collision. Unfortunately, for non-interruptible collision handling systems, delays are one of the more likely side effects when there is a sudden increase in computational workload, such as when a large number of objects happen to collide simultaneously.

To evaluate the effect of such delays, an experiment was set up to extend Michotte's studies for animated 3D scenes. Twenty participants were recruited ranging from 19 to 35 years in age. These were chosen from staff and students of Trinity College and each had some knowledge of computer graphics but little to no qualified background in dynamics or kinematics. Participants were shown animations of collisions involving very simple objects: two spheres of equal volume and mass colliding and moving apart (see Figure 6.4). The collisions all occurred in the centre of the screen. In each simulation run the relative initial and final velocities of the objects were equal and opposite but delays of 0 msec, 100 msec and 300 msec were artificially introduced at the instant the objects collided. For each delay time the simulations were run for 3 different initial velocities and from 3 different viewpoints, with 3 replications of each condition. Hence a 3x3x3 experiment replicated three times involving a relatively simple rating task. Participants were asked to rate each collision on a simple integer scale of 1-3, where 1 was very believable and 3 was very unbelievable. They were subsequently questioned about their strategy.

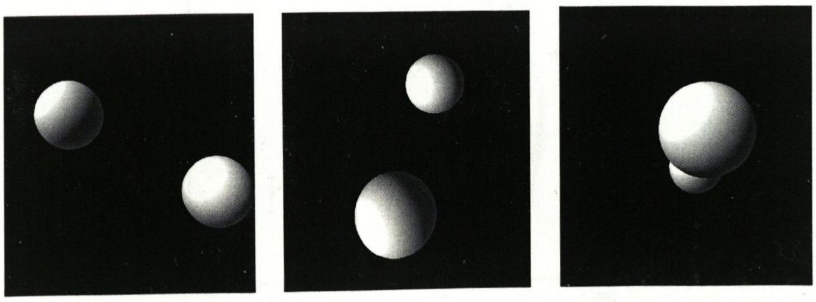


Figure 6.4: Causality Experiment in 3D

Results

We found that the effect of delay upon the rating given to collisions was highly significant: > 99%, as can be seen in Figure 6.5. Please note that low scores are better. We examined the data collapsed over viewpoint and velocity and found no significant effect. The delay effect was also significant when examined for each velocity separately. In addition, almost all participants complained about the collisions that seemed to "stick together" or were "less bouncy".

Analysis

These result are consistent with Michotte's, in that the addition of a delay reduced the perception of causality, thus impacting negatively upon collision realism. We might therefore conclude that the longer a real-time system spends processing collisions and the longer the delay that is thus generated, the less believable the resulting collisions will be.

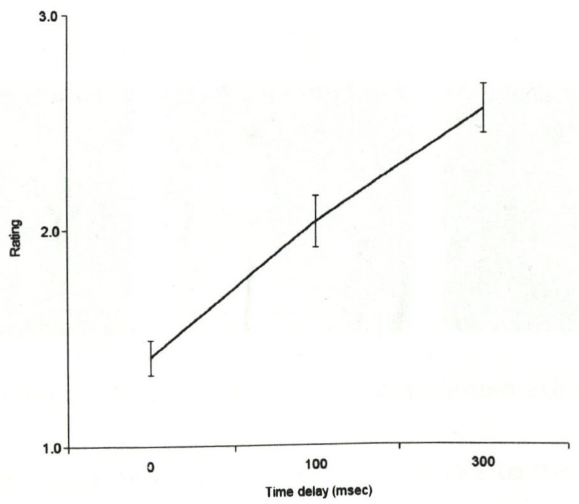


Figure 6.5: Effect of an induced delay on the rating of collisions (low scores are better)

6.2.2 Dynamics and Kinematics

Directly following the causality experiments, two experiments were performed with the same 20 participants to test their sensitivity to variations in the levels of collision detail. The first experiment tested the impact of varying levels of the volume model used for collision detection. As described in Section 6.1, lower levels of sphere-tree detail result in larger gap sizes in collisions. Participants were shown collisions involving three-dimensional L-shapes (see Figure 6.6). These simple concave extended bodies were chosen as they are useful for testing the features of the collision detection system. The starting velocities and orientations were set so that the resulting collision at the centre of the screen caused both objects to be repelled back in directions directly opposite to their initial velocities. Users were again asked to rate the collisions on a simple scale of 1 to 3 and were questioned on their strategy afterwards. The simulations were run at varying levels of sphere-tree detail from 3 different points of view and with the objects' velocities being scaled by factors of 50%, 100% and 150%, with 3 replications of each condition; once again a 3x3x3 experiment with 3 replications.

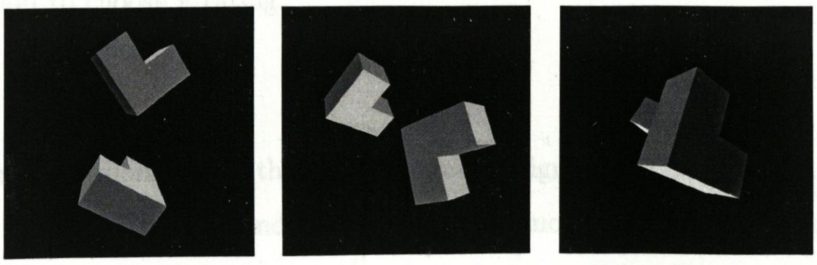


Figure 6.6: L-shaped bodies used for dynamics/kinematics experiments

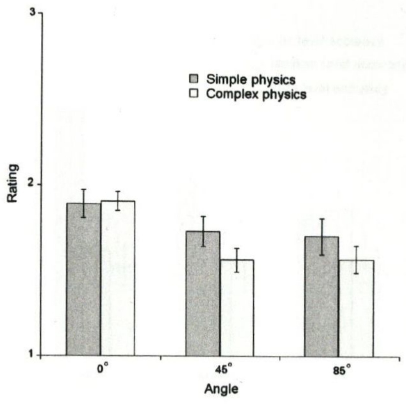
The second experiment tested participants' responses to the model for collision response used in the system. As we stated in Section 6.1, varying the levels of collision detection detail has a direct effect on the objects' final velocities after collision. Participants were shown collisions involving the same L-shaped bodies used in the previous experiment. This time however the objects started off with different initial conditions with the resulting collision causing both a change in trajectory and angular momentum. The main goal of this test was to evaluate how adversely the collision levels of detail affected the perceived collision response. The lowest level of collision detail involves approximating the objects as spheres. As a consequence of this, the calculated collision impulses tend not to impart changes to the angular velocity of the objects. Because such response looks so unbelievable, as one expects rotations in collisions involving such non-symmetric objects to result in spin (an observation later verified by user feedback), a random change was added to the objects' angular velocities at the lowest collision level. This change was uniformly distributed between two limits that were chosen by inspecting the values generated for a series of accurate collisions.

One hypothesis we tested was whether the viewpoint affected the perception of the collision anomalies. There were three cases, the first being the 0° case, where objects approached each other parallel to the viewing plane. The other cases involved angles of 45° and just less than 90° respectively. In the latter condition, although the viewers could not see the actual points of impact, they could see the object as it approached and bounced off, thus providing them with enough information to choose a rating.

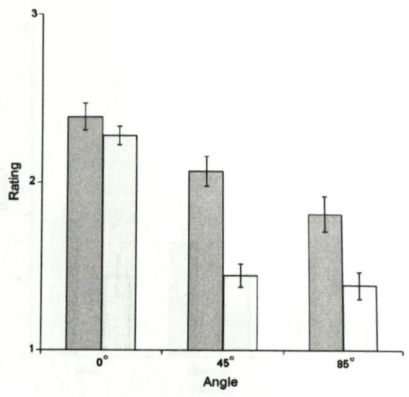
Results

In the simple response case, the results were not significant: $> 75\%$, with a more significant effect in the second experiment with more complex physics: $> 99\%$ (see Figure 6.7a). In particular we were interested if the obscured viewpoints reduced the number of gaps perceived. We found that the main effect was with the largest gap size only, *i.e.* lowest level collision handling, where the effect was highly significant for both experiments but particularly strong for the more complex physics (see Figure 6.7b).

We also predicted that velocity would reduce a viewer's ability to detect anomalous collisions. We looked at the overall effect for both experiments and discovered that there was only a weak effect in the simple case, where perception was



(a) Overall effect



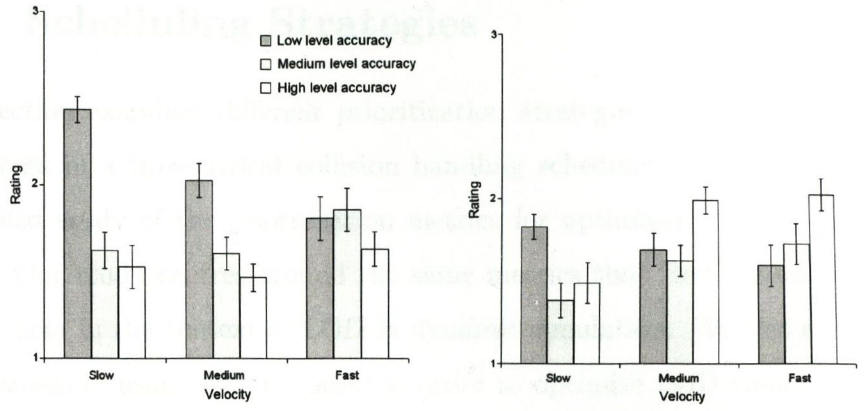
(b) Low level collisions only

Figure 6.7: Effect of viewpoint on rating of collisions

worst for the slower collisions, but there was an opposite and more significant effect in the complex physics case, where faster collisions were actually rated worse. These unexpected results led us to examine the effect of velocity at each level of detail separately. We found that at the slowest velocity, the effect of collision resolution was most significant in both cases, with the effect being most significant in the experiment with simple physics. In this case, performance was more as expected (see Figure 6.8(a)), with low resolution collisions being less detectable with increasing velocity.

The results in the experiment with complex physics were more surprising (see Figure 6.8(b). When velocity was slow, the low-resolution collisions were more obvious and hence the worse rating. However, as velocity increased, participants actually rated the more accurate collisions worse than the medium level and those with randomised low-level responses.

Figure 6.9 shows the effect of the different collision resolutions for both experiments. In the simple physics case, the effect of low resolution detection is highly significant: $> 99\%$, whereas in the complex case, the effect is less significant: 90% and we might conclude that we have quite effectively masked the negative impact of reduced physics in the lowest resolution case by introducing a completely ran-



(a) Simple physics (b) Complex physics

Figure 6.8: Effect of velocity on rating of collisions

dom rotation after contact. This was confirmed by the comments of almost all participants who reported that the most realistic collisions were those that spun a lot after colliding.

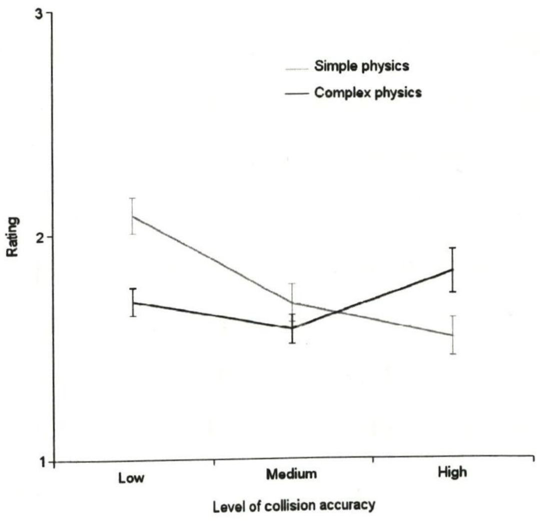


Figure 6.9: Overall effect of collision detection accuracy

6.3 Scheduling Strategies

This section examines different prioritization strategies and compares their effectiveness in a time-critical collision handling scheduler. Reddy [81] provided a detailed study of the prioritisation metrics for optimising animation levels of detail. Our study centres around the same metrics that Reddy uses but we examine them in the context of LOD in dynamic simulation. We also examine the effectiveness of using an eye-tracker in order to optimise LOD trade-offs in a 3D dynamics scene.

A simple study was done to compare the different strategies discussed in 5.2.1 for prioritising a dynamic scene for collision processing. As we mentioned previously, assigning thresholds for discrete partitioning of the scene into priority groups is still a very subjective issue. Many factors can affect what the ideal thresholds would be for partitioning the scene and as explained in Section 5.2.1, the optimum threshold is very specific to each scene and in most applications such thresholds are chosen after some sampling of the scene to statistically determine an ideal value. Thus we do not offer the following results as a conclusive comparison of the effectiveness of a partitioning scheme but as an illustration of the different characteristics of the different schemes we have implemented.

In each case, a simple scene was taken with 80 uniformly sized objects moving within an enclosed scene, starting at random positions and with randomised starting velocities (see Figure 6.10). A perspective projection was used and objects were reflected back into view at the window boundaries (*i.e.* they were enclosed in the viewing frustum) in order to preserve the density of objects in the scene. The thresholds for partitioning were chosen *ad hoc* with the proviso that close to the same ratio of high priority and low priority collisions should be preserved in each test run.

A series of scheduling functions were used to partition the collision list into a High Priority List (HPL) and Low Priority List (LPL). The simulations were

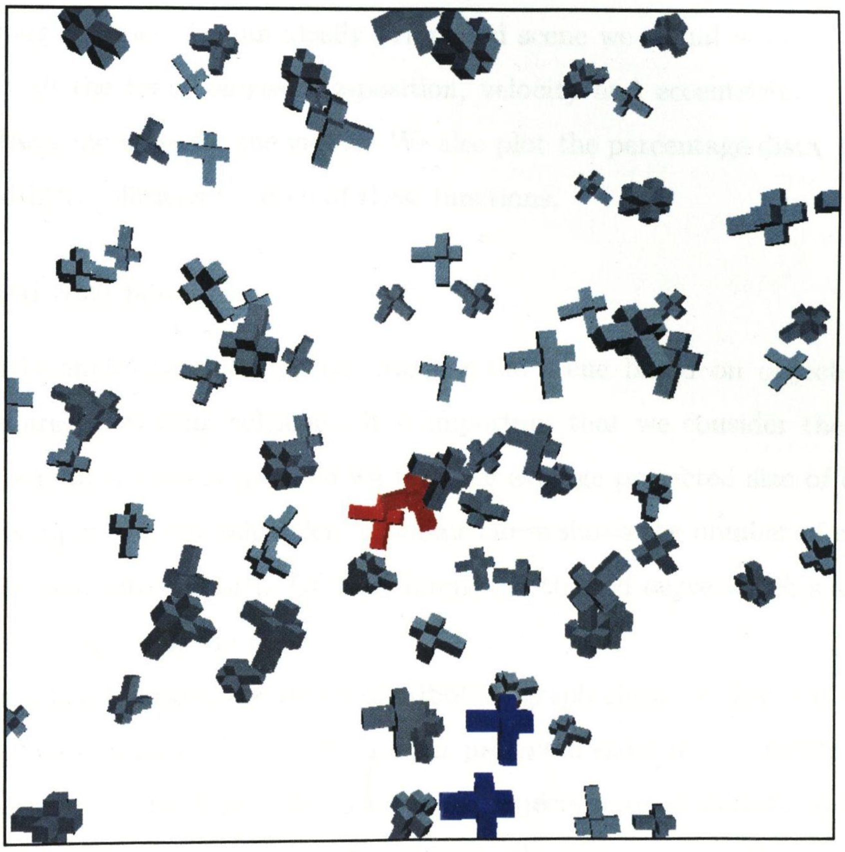


Figure 6.10: Test Application

run for 10 minutes each and the total collision instances entered into the HPL and LPL were counted. To illustrate the distribution of priorities across the scene due to each scheduling function, we plot (for each run of the experiment using a particular scheduler) the distribution of High and Low priority collisions according to z-axis distance, velocity distribution and eccentricity distribution. The reason we choose these three distributions is because they are representative of traditional partitioning schemes. For an ideally prioritised scene we would wish to see high values at all the lower ranges of z-position, velocity and eccentricity, fading as each of these increases for the viewer. We also plot the percentage distribution of HPL and LPL collisions for each of these functions.

Projected Size Scheduler

Figure 6.11 shows results from partitioning the scene based on objects' sizes. Since we are prioritising collisions, it is important that we consider the size of both objects in a collision pair, so we take the average projected size of the two objects as input into the scheduler. The blue curve shows the number of collision objects entered into the LPL. Of more interest is the red curve which shows the number of collisions in the HPL.

As we should expect, the z-axis distribution graph shows a clear cut off at a certain z-axis distance. This is because the projected sizes of the objects are directly proportional to their z-depths since the objects were of uniform size. Thus we are effectively using the z-axis to partition the scene. We might expect the speed and eccentricity graphs to be largely random, however the percentage HPL distribution increases with speed due to larger objects (placed in the HPL by the size scheduling) having larger projected velocities due to perspective projection. The total number of collisions (for both high priority and low priority lists) increases with eccentricity for similar reasons, simply because increasing numbers of distant objects fit on the screen due to the perspective projection.

Speed Scheduler

Figure 6.12 shows the results of a scheduler that uses projected speed as the prioritisation function. Once again we take the average speed of the two colliding objects. Higher velocity object pairs are given a low priority since it will be harder to detect anomalous collisions at high speed. This is based on our studies in Section 6.2 and the results of Reddy's model of fall-off in perceptual significance with velocity [81]. It should be noted that Reddy reported that if velocity is the only factor the velocities needed to be considerably high before the perceptual impact of objects begins to fall. However, in tandem with other factors such as eccentricity, the effect occurs at much lower velocities. The HPL distribution along the z-axis increases with z-distance because the projected speed naturally decreases with distance due to the perspective projection. This is the inverse of what we stated for velocity in the size scheduler and is somewhat undesirable as we really want collision priority to decrease with distance. As expected we see the cut off at the threshold value in the speed distribution graph. For eccentricity, we see increases in the general number of collisions with distance due to perspective as before, so the percentage graph is a more meaningful measure in this case and we see from this that eccentricity distribution in the HPL is relatively unaffected by the velocity metric.

Eccentricity Scheduler

Figure 6.13 shows the results of partitioning based on eccentricity. We take the image space distance between the centre of the screen and the projected centre of collision as input into the scheduler. HPL entries are fairly constant in both the z-distance and velocity graphs. The percentage graph shows that velocity distribution is fairly random due to the fact that eccentricity scheduling does not affect HPL distribution sorted by "velocity. The fact that the camera used in this experiment was stationary is an important point to note. Had we used a moving camera, the full set of variables in Equation 5.1 would apply and we should expect the eccentricity and velocity metrics to have some interdependency.

Gap Scheduler

Figure 6.14 shows distributions due to scheduling according to the gap size[1] between the two colliding objects. Z-axis distribution is random as this is an object space scheduling strategy. We might expect some correlation with velocity as high velocities tend to cause deeper interpenetrations (and hence smaller gap sizes) but this is not evident from the results. As expected this has no bearing on the eccentricity distribution which is largely an image-space metric.

Projected Gap Scheduler

Figure 6.15 shows distributions due to projected gap size scheduling. Projected gap is simply the separation distance used in the gap size scheduler projected on to the image plane. We can see a fall off in HPL collisions along the z-axis which is both useful and expected due to the perspective projection reducing projected gap sizes. In the speed graphs we see increasing numbers of HPL entries at high speeds. This is due to high velocity causing deeper interpenetrations (and as a result smaller gap size). The effect is now clearer because we are using image space measures for both speed and gap. There is a small rise in the eccentricity distribution.

The increase in HPL entries with speed is not desirable if our hypothesis (supported by our results in Section 6.2) is that high velocities collision hide anomalies and thus should be placed at a lower priority. The small rise in eccentricity distribution is unexpected as this suggests that projected gap size increases statistically with eccentricity. This may be a consequence of the enclosed scene or of the nature of the nature of perspective projection which we have not investigated yet

[1]A coarse approximation for potential gap size taken here is the Euclidian distance between the centers of the two colliding bounding volumes

but since the since the increase is small it is not on its own significant enough to invalidate the usefulness of the projected gap scheduler. In fact, projected gap would seem to be the best metric so far, as it incorporates z-distance prioritisation and possible occlusion as well as the collision gap size which is important on its own.

Collision Speed

Projected speed is taken as the projection of the relative velocity vector between the two colliding objects. Lower projected velocities are given a higher priority and this is graphed in Figure 6.16. In all three cases we see no significant effect although there is a small increasing trend with z-distance as relative velocities of distant colliding objects are mapped onto smaller projected speeds. Conversely there is a small decrease with projected speed as the two factors are inter-related.

Discussion

This study evaluated some characteristic properties of different scheduling strategies. We find that not only do we find that several metrics work independently of others but that some of them actually return results that may contradict another sorting scheme *e.g.* the velocity scheduler increases HPL entries with distance from the viewer.

In terms of immediate use, we favour metrics that take collision variables into account whilst preserving the behaviour of the more generic prioritisation techniques of distance, velocity and eccentricity. We also recommend spatially coherent sorting as it is easier to safeguard against popping. For instance, a distance scheduler will show a clear boundary between where HPL and LPL collisions occur, making it easier to predict which objects are likely to exhibit popping effects, namely the ones that are about to leave the high priority region. This is unfortunately not so for factors such as scheduling based on collision velocity, as

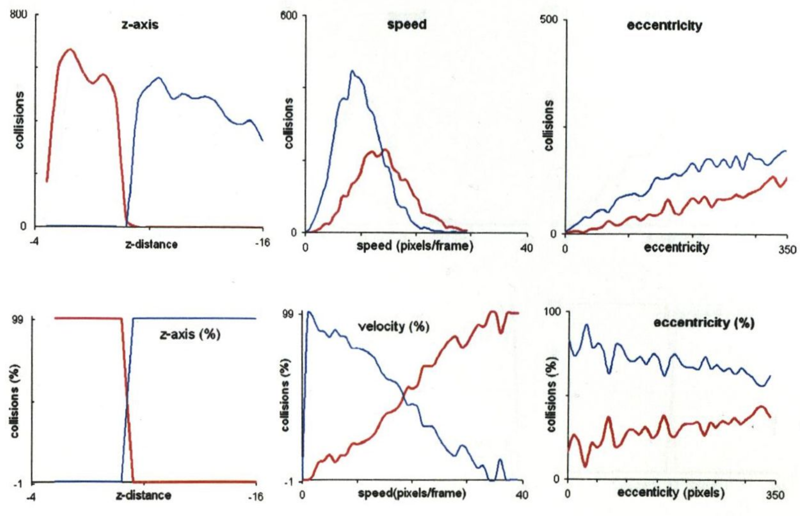


Figure 6.11: Priority distribution due to the Size Scheduler

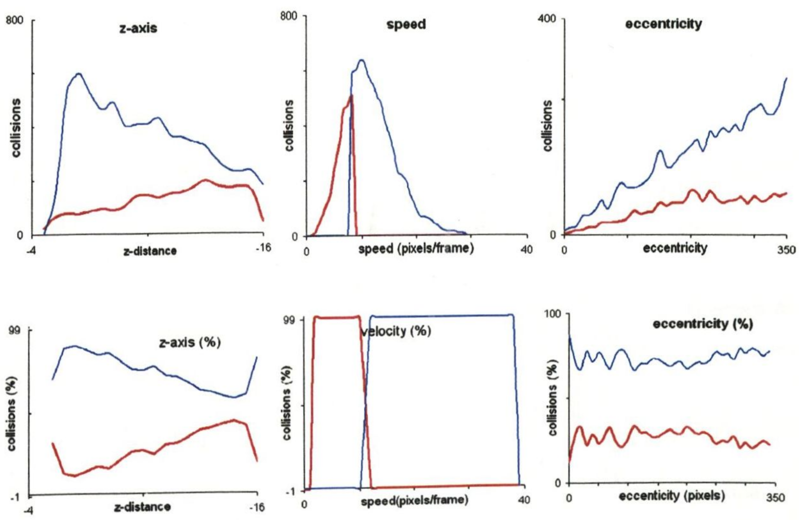


Figure 6.12: Priority distribution due to the Speed Scheduler

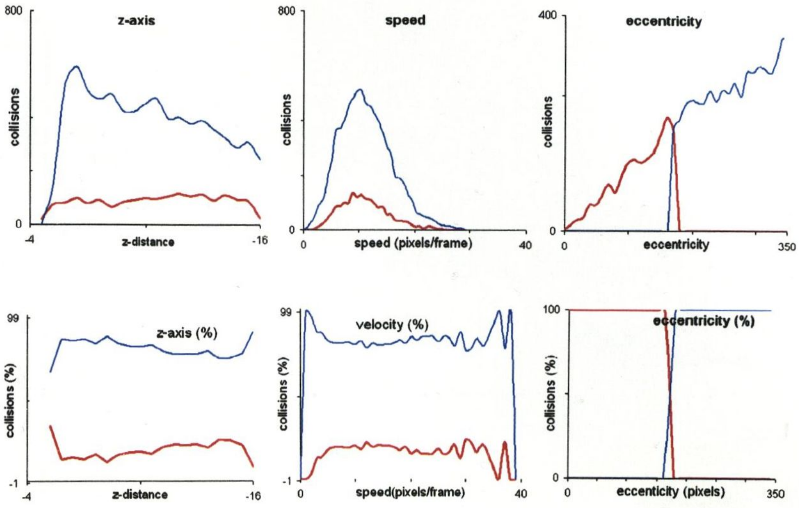


Figure 6.13: Priority distribution due to the Collision Eccentricity Scheduler

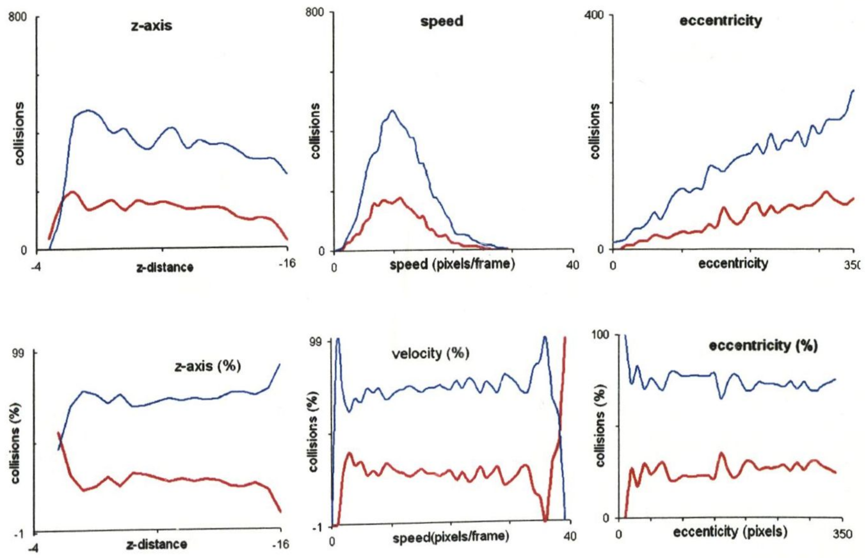


Figure 6.14: Priority distribution due to the Collision Gap Scheduler

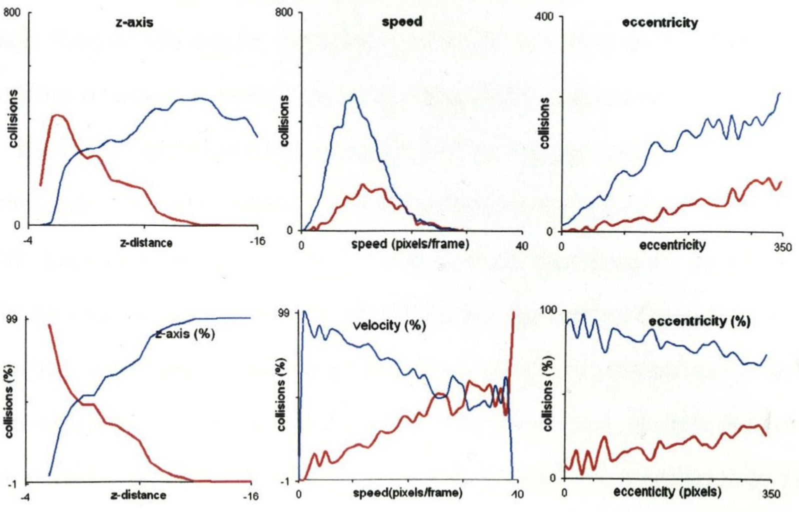


Figure 6.15: Priority Distribution due to the Projected Collision Gap Scheduler

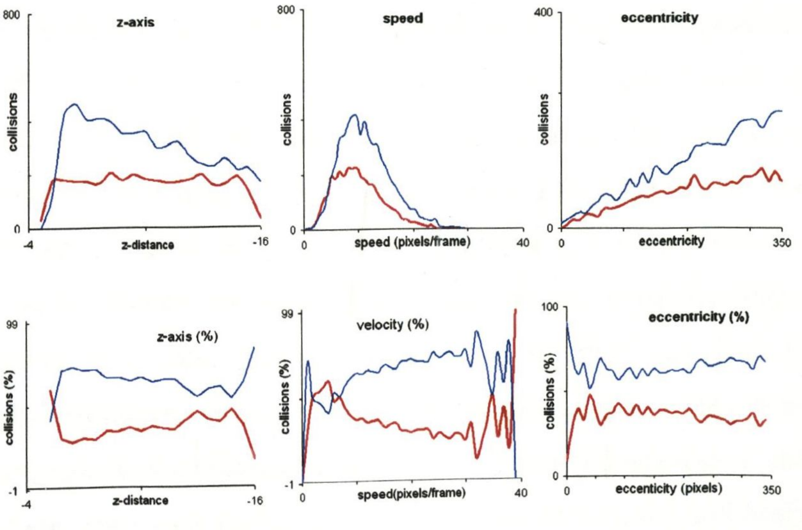


Figure 6.16: Priority Distribution due to the Projected Collision Speed Scheduler

the HPL entries are scattered throughout the screen and are not dependent on the state of a single object but possible combinations of collision pairs.

Although simple efficiently evaluated metrics are desirable, there is a strong indication that a single measure is not completely adequate for a time-critical scheduler. However, as O'Sullivan showed [74], we must take care not to overburden the scheduler with too many complex calculations lest it become a hindrance to real-time performance on its own. The correct balance of input factors and LOD resolution time can only be determined by extensive further study. This is a question that needs to be answered in the context of perceptual science and a thorough investigation unfortunately is beyond the scope of this thesis. Current and future work is planned however in this respect and we discuss this in Chapter 7. We also provide a specific case study of a gaze dependent scheduler in the next section.

6.3.1 Perceptual Optimization

To further evaluate the effectiveness of a gaze-dependent prioritisation scheme for interactive simulation, an experiment was performed. Ten participants (computer science staff and students) were presented with 36 short simulations of rigid bodies colliding and bouncing off each other inside a closed cube. The simulation was run on a desktop PC with graphics acceleration, with a 22-inch screen. Participants were instructed to react to the quality of the simulations in two different ways. The first task was to respond, by clicking the mouse button, whenever they perceived the occurrence of a frame containing one or more "bad" collisions during the course of the simulation. A bad collision here refers to one resulting from a coarse level approximation of a collision as described in the previous section. In an initial training phase, they were shown examples of what both good and bad collisions should look like. The second task was to rate the overall quality on a scale of one to five, at the end of each simulation. During the training phase, examples of the best and worst quality simulations were shown, and they were told that the two extremes should receive a rating of five and one respectively. They were also told that simulations with quality ranging between both of these limits were also possible. They then practised on a further number of simulations and were observed to ensure that they had understood the instructions.

Four distinct types of simulations were presented to the participants in random order. The first type of simulation (denoted as *all good*) resolved all collisions at the highest resolution of the volume model. It was possible to deal with objects at this high a resolution in the experiment as the maximum number of objects dealt with was relatively small. A second type of simulation (*all bad*) dealt with all collisions at the very lowest level of resolution *i.e.* object collisions were dealt with at the bounding sphere level, resulting in objects repulsing each other at a distance in almost all cases. It should be noted that this distant repulsion is not always obvious to viewers as inter-object occlusion sometimes prevents the gap from being visible in the projected display.

Two further types of collisions had combinations of good and bad collisions occurring in the scene at the same time throughout the simulation. In both of these, a *high-priority region* was chosen in the scene where collisions were dealt with at the *all-good* level while outside of this region all objects were dealt with at the coarse level. In one of these, the *tracked* simulations, the users gaze position was tracked and used as the centre of the high-priority region. In the other case, the *random* simulations, a random position was chosen every 5 frames to serve as the centre of the high priority region. Having a randomly located priority region of the same size in the scene ensures that roughly the same proportion of good and bad collisions is maintained as in the *tracked* case.

Each simulation type was shown with 5, 10 and 15 objects and three repetitions each were shown for these twelve cases, see Figure 6.17. Hence the overall experiment had a 5x3x2 factorial layout with three repetitions of each case. A time of ten seconds was chosen as the duration for each simulation to give the participants

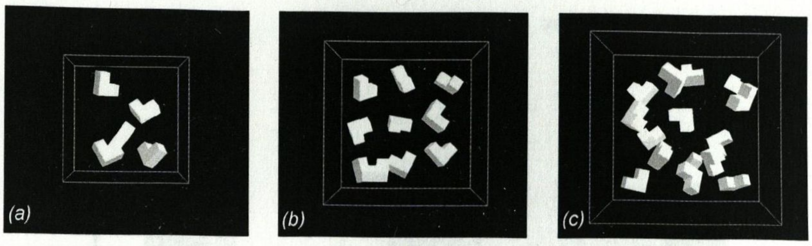
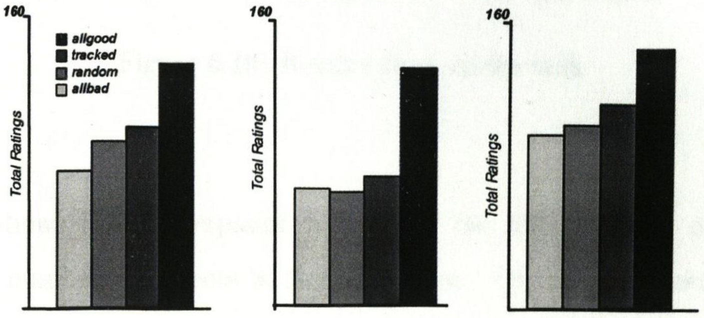


Figure 6.17: Screenshots of the experiment with 5, 10 and 15 objects contained in proportionately-sized boxes.

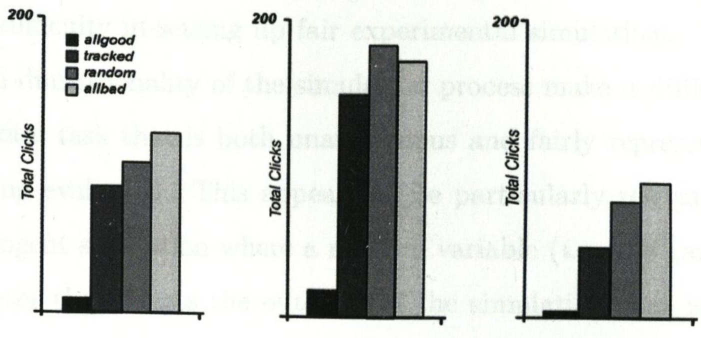


(a) Five objects *(b) Ten objects* *(c) Fifteen objects*

Figure 6.18: Results from rating task

a representative sample of collisions. In varying the number of simulated objects, the size of the cube, within which the objects were contained, was correspondingly resized to maintain a constant density of objects at all times within the container. This was in order to ensure consistency in the number of collisions occurring in the simulation. The size of the boxes displayed were not scaled to fill the screen, as then the size of the objects would vary between conditions. Necessarily, this reduced the active field of view for the smaller number of objects.

After a short training phase, in which participants were shown isolated cases of good and bad collisions, participants eye-movements were recorded at all times with an SMI EyeLink eye-tracker and the 36 simulation runs were shown in random order.



(a) Five objects *(b) Ten objects* *(c) Fifteen objects*

Figure 6.19: Results from clicks task

Results

Figure 6.18 shows the participants' ratings for the different sets of simulations organised by number of objects in the simulation. Of most interest here, in the context of a gaze-contingent system, is the comparison of the *tracked* and *random* graphs. The results clearly show an overall improvement in the perception of the *tracked* simulation. A single-factor ANOVA showed a significance of > 70%, > 55% and > 80% respectively for the 5, 10 and 15 object results.

An examination of the number of clicks for each of the simulations shows similar results. There is an overall improvement in the number of clicks in each of the simulation cases with statistical significance of > 60%, > 75% and > 80% respectively for the 5, 10 and 15 cases. The graphs of the total clicks during the simulations, shown in Figure 6.19, show a consistent reduction in the number of clicks for the 15 object simulations. It is reasonable to assume that this is due to an increase in the number of occluded objects as well as in the number of similar distractors as discussed in Section 2.3.1.

Discussion

It was surprising to find that the results showed little or no statistical significance. We believe that the reason that there isn't a stronger significance has to do with the difficulty in setting up fair experimental simulations. The complexity and multi-dimensionality of the simulation process make it difficult to design an experimental task that is both unambiguous and fairly representative of the variables being evaluated. This appears to be particularly relevant in the case of gaze-contingent simulation where a random variable (*i.e.* the gaze position) is an active factor that affects the outcome of the simulation that is given to the participants for evaluation. Modulating simulation detail levels at random or gaze-dependent locations in the scene introduces a significant level of non-determinism into the simulation making it close to impossible to show all the participants an identical set of simulations. Future work is planned on investigating alternative experimental strategies.

Some participants reported that they used peripheral vision in certain cases to decide on their rating for the simulation. The experiment used a simple, two-level scheme for prioritisation (*i.e.* fine resolution within the high-priority region and coarse resolution everywhere else). While there are some studies that examine the ideal size of high-resolution regions for scene viewing, as in [61], there is no documented study that suggests an ideal radius for a high priority region for simulation purposes and it is expected also that this would be a subjective value for each scene.

6.4 Sample Output Screenshots

Sample screenshots from a typical application are shown in Figures 6.20, 6.21 and 6.22. The objects we have chosen to simulate are polygon meshes with large concave features and high polygonal detail. It should be noted that the number of polygons has no direct bearing at run-time on our collision handling system. While most exact methods would use the mesh polygons, or a simplified version of the mesh to perform collision tests, our system would use the mesh for generating the volume model at a pre-computation phase [7, 8]. The scene shown here is one with increasingly large numbers of detailed concave objects moving in an enclosed space (a cube) which forces many simultaneous collisions to occur.

An eccentricity based scheduler was used and the high priority collisions have been flagged in red; low priority collisions are in blue. Even with the huge amount of simultaneous contacts, the system was able to continue in real-time with a frame-rate of 20 per second. In fact, as we increase the number of objects we find that the system begins to slow down sooner due to rendering. It should be noted that many collisions would be dealt with at a very coarse level and theoretically we should be able to handle as many objects as it is possible to do bounding sphere collision tests for. However, due to the crowdedness of the scene, these anomalies become less apparent as the complexity increases.

6.5 Summary

This chapter discussed the results of the work described in earlier chapters. We have presented output from our adaptive collision detection system and shown how collision data is refined with increasing processing time. Unfortunately, due to the unique approach we have taken, it is difficult to compare the technique with existing collision handling systems which are based on highly different goals of accuracy and exactness. A meaningful measure of the effectiveness of an adaptive system, such as the one we have developed, is only achieved through perceptual evaluation. However, the science of perceptually evaluating graphical animated scenes is itself in its infancy and no established approaches exist to quantitatively evaluate physically based animation systems. Thus, we have presented results from our own qualitative evaluations performed through user tests based on the perception literature. From these we outlined some key factors that affect the perceptual plausibility of simulation.

We have also presented, in this chapter, a comparative evaluation of different scheduling strategies discussed in Chapter 5 and further user experiments to judge

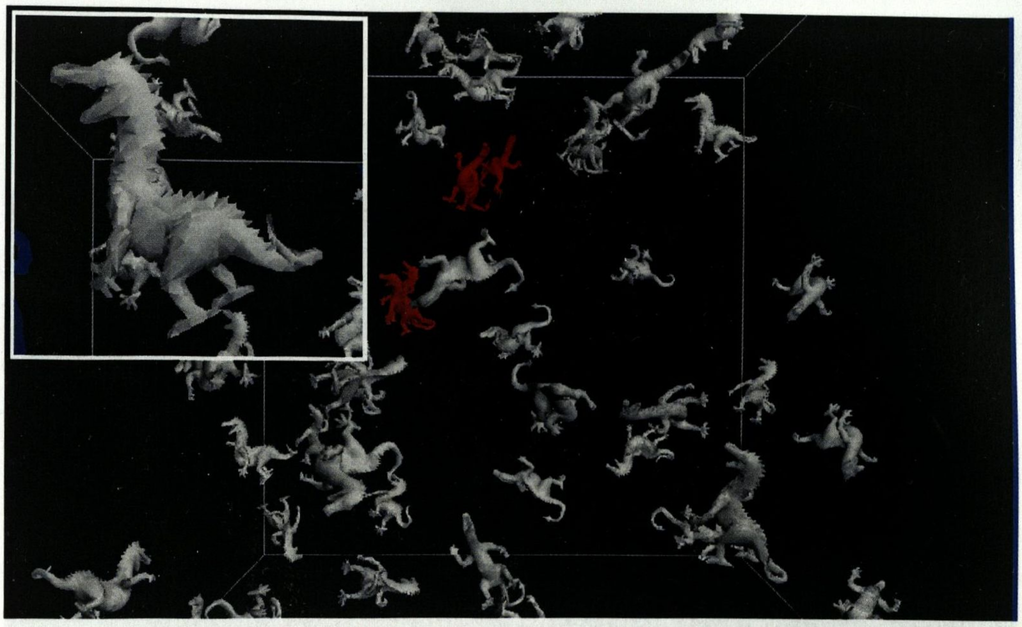


Figure 6.20: 50 Dragon Simulation

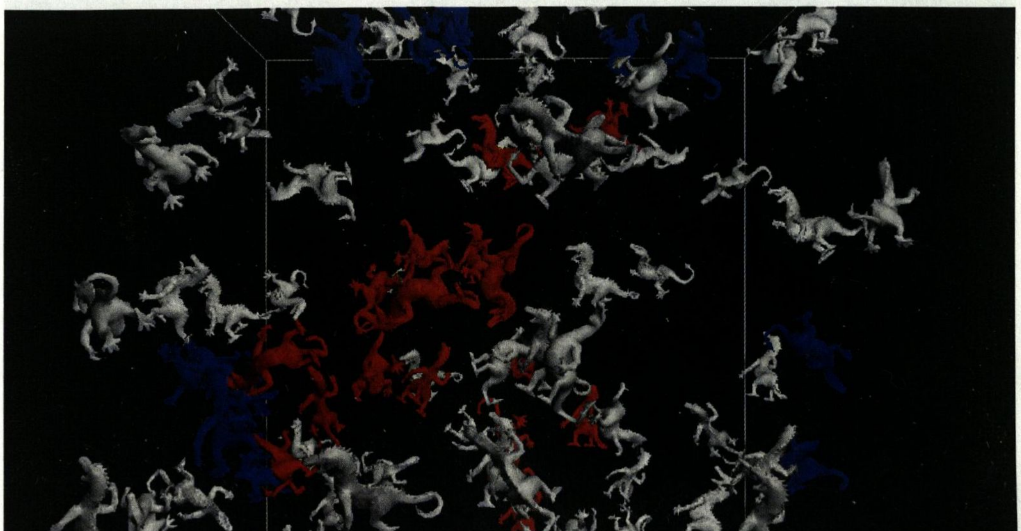


Figure 6.21: 100 Dragon Simulation

the effectiveness of a gaze-directed scheduling system. Lastly, we presented output screenshots from sample scenarios that traditional collision handling systems have difficulty dealing with and that our system is specifically optimized to handle.

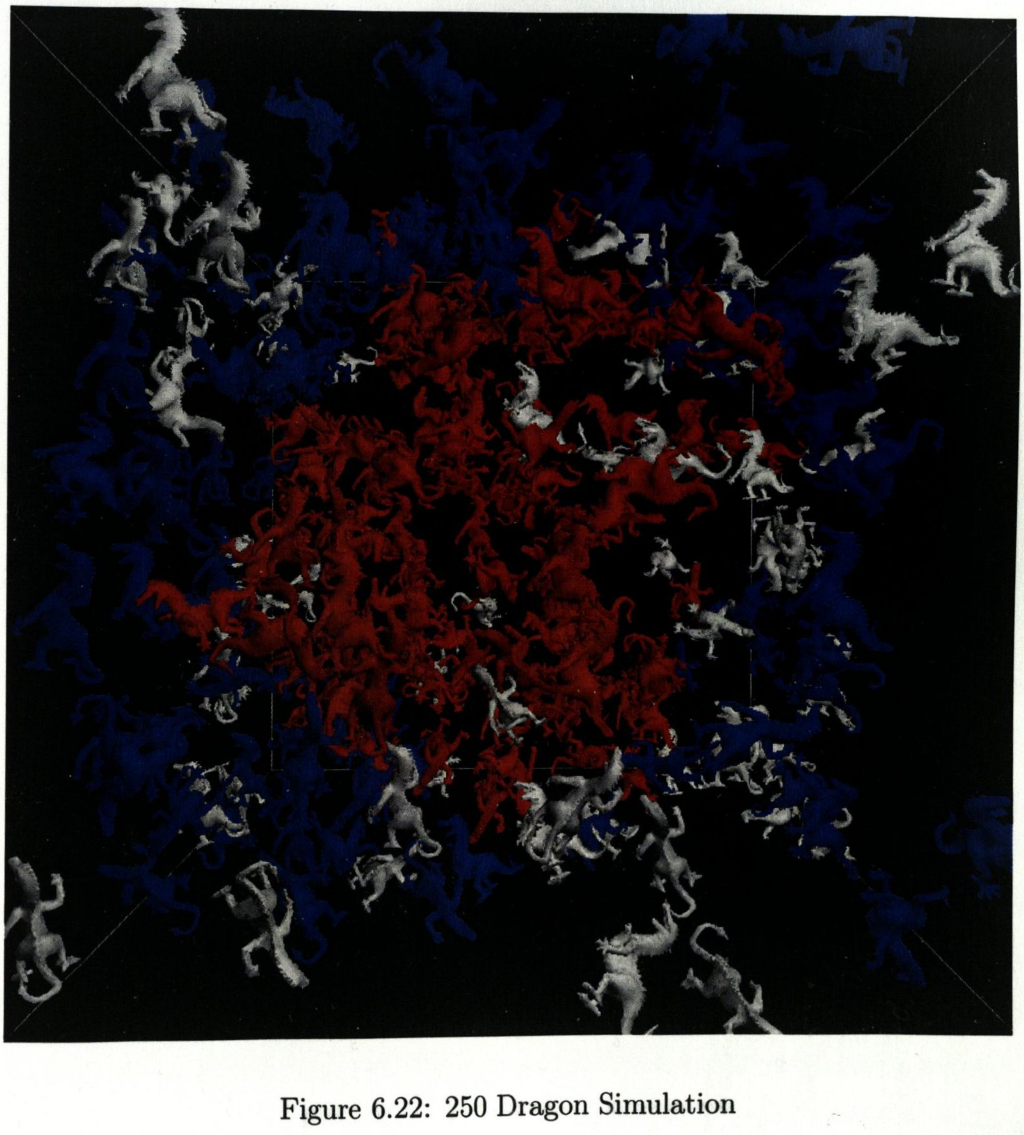


Figure 6.22: 250 Dragon Simulation

the effectiveness of a gaze-directed scheduling system. Lastly, we presented output screenshots from sample scenarios that traditional collision handling systems have difficulty dealing with and that our system is specifically optimised to handle.

Chapter 7

Conclusions and Future Work

This section reviews, in brief, the findings presented in this thesis and the solutions that have been offered. We also discuss new questions that have been raised due to the work presented here and some potential areas of future work.

7.1 Summary of Contributions

The main goal in this thesis has been to provide a time-critical framework encompassing all three stages of collision handling. We have presented such a framework, in Chapter 3, which performs time-critical collision detection based on the sphere-tree approach by Hubbard. We add our own mechanisms for sphere-tree contact modelling and for scheduling the full process of collision handling. In Chapter 4 we provide further optimisation to the contact modelling phase by using heuristic contact primitive reduction techniques in order to reduce the workload of collision response without excessively sacrificing the detail that collision detection and contact modelling are able to achieve. We also describe additional bounding volume hierarchies that can be used under the sphere-tree based framework in Chapter 3, including an SSV based hierarchy used for collision handling of articulated figures, and present a framework for collision detection and contact modelling with heterogeneous BVH trees. Chapter 5 describes optimisations to the scheduling process in Level of Detail collision handling. We discuss existing strategies for prioritising animated scenes and adapt these for the specific case of collision handling in dynamic simulations. We also discuss problems unique to level of detail in simulation and solutions to these. Chapter 6 details some of the results of our work, including the Level of Detail output from the contact modelling mechanism as well as psychophysical tests to explore user sensitivity to collision anomalies in a virtual scene. We also compared different scheduling strategies presented in the early chapters and performed user tests to validate a gaze-driven approach to scheduling collision handling.

7.2 Assessment

Although an actual implementation was developed during the course of this research, the primary goal has always been to cultivate a general philosophy of time-critical dynamic simulation and prove the feasibility of using approximate and refinable methods in an area that has traditionally dealt with exact solutions. We have presented, in this thesis, an in-depth study of the time-critical approach to collision handling, inspecting all of the key required components for such a system to be fully realised. We have highlighted some problems that such systems may face and developed and documented some solutions to these.

As we present a distinctly different type of solution to what previous approaches have attempted, the difficulty lies in evaluating the relative effectiveness of such a solution. Very few agreed-upon criteria exist for comparing even existing exact collision detection methods apart from computational speed, accuracy in detection and the limits in scene complexity. The system we describe, by its very time-critical nature, makes timing comparisons with other exact approaches inappropriate. We are able to deliver multi-resolution scenes of adaptive complexity and with regards to accuracy we maintain that in the interactive applications, for which this approach is designed, it is more relevant to measure the plausibility and perceptual impact of the output.

Thus, we propose that such systems be evaluated on the basis of perceptual criteria and present some background on how this might be reliably achieved using models of human perception. Unfortunately, the science of perceptually evaluating computer graphics is still relatively in its infancy. Few established approaches exist which generically and reliably are able to return a qualitative evaluation of a graphical scene. Fewer still exist for evaluating interactive animations and to our knowledge no documented approach exists to date for measuring the perceptual impact of simplifications in generic complex dynamic simulation, although O'Sullivan [74] investigates collision anomalies and Reddy provides a low level study of perceptual factors to do with virtual scenes [81]. Based on previous studies in perceptual science, on determining human response to dynamical events, we present results from our own user-based experiments to evaluate the perceptual impact simplifications have on simulated dynamics. We demonstrate that the factors which we have sought to optimise using our approach for adaptive collision handling, are indeed important in the effect they have on plausibility of dynamic simulations. The issues of causality, separation and dynamic accuracy are shown to have significant impact on user response to the system and thus it is relevant to address the problem of simplification with these factors in mind. This is consistent with the framework that we have presented in this thesis.

7.3 Future Work

The use of perceptually adaptive techniques in computer graphics is relatively new. It is only with the recent advance in computational power that graphical output from interactive real-time applications has begun to approach the quality that we know is possible through traditionally off-line approaches. Yet, it is likely that it will take a few more stages of evolution in computer graphics hardware before it becomes possible to deliver perceptually identical output in off-line and real-time graphics. Until then it seems likely that adaptive techniques will be used in various aspects of real-time graphics systems and indeed there appears to be an increasing emphasis on research in level of detail, time-critical and perceptually adaptive techniques not only in image synthesis but in animation and simulation.

However, the introduction of these new adaptive techniques and more generically the philosophy of perceptual adaptiveness in real-time applications opens up a Pandora's box of possible problems and opportunities for future optimisation. We can draw many analogies between level-of-detail in physically based motion synthesis and its more-established use in real-time rendering. Among the problems that have only recently been addressed in rendering and need still to be tackled in motion synthesis are the points discussed in the remainder of this section.

Effects of Popping in Simulation

As level of detail in simulation becomes more evolved, we see many problems that are common to LOD in rendering. One of these is the popping problem which has been discussed in this thesis in the context of Collision Handling. More study needs to be done in assessing the full impact of popping in the general context of simulation and in devising useful reusable solutions such as have been implemented for rendering levels of detail.

Generic Metrics for Dynamic LOD

Some well established methods exist for qualitative assessment of rendered images. The study of metrics for animation has also been investigated by a number of researchers [30, 74, 81] but in some of these cases the results are very specific to the problem that the approaches were addressing. Research to date has been more to do with exploring trends and relationships rather than establishing unbiased thresholds and models that practitioners can use in their real-time systems. There is an added difficulty in evaluating the perception of animations due to the high number of variables involved and due to the intangible nature of what is being perceived by the users. Thus it is a challenging task simply to understand and interpret with certainty why users have responded favourably or negatively to any particular dynamic scene, let alone to derive meaningful metrics from user tests. Our approach so far has been to try to extract basic low level variables and evaluate user response based on small perturbations to these. Many more low level variables remain yet unexplored, which range from visual factors such as colour, luminance, and contrast, to attributes of the dynamic system itself such as orientation, mass and density. Work is currently underway to measure the impact of some of these factors, which we believe are of primary concern in a dynamic simulation. With further examinations and study, it is hoped that we will be able to build, from these low level models, more encompassing metrics for a highly complex dynamic scene. Ensuring that these are generically applicable will be a challenging task but more research in the area is essential in order to make full use of adaptive level of detail techniques.

Generic LOD Resolver

Assuming we have reliable measures to evaluate perceptual quality, these should logically be incorporated into level of detail schedulers in adaptive systems. This may not be feasible in some cases where evaluation of the metric is expensive and prohibitive to the real-time performance of the animation system, but quicker methods, that take into account low level perceptual factors, would be useful for managing level-of-detail modulation in virtual scenes.

With the complexity of current and upcoming virtual environments, animation systems are increasingly implementing LOD techniques for various aspects of the animation systems, including the systems for rendering, behavioural simulation and interaction. Where adaptive LOD is being used it may be feasible to suggest that all of these process should be managed by a single scheduler which allocates computational resources to the different systems. We discussed in the context of dynamics simulation how even the simple question of threshold values can be extremely subjective. Thus in order to be applicable generically to animation systems, a Level of Detail Resolver needs to be both efficient as well as flexible and reconfigurable for different problem specifications.

Generating Bounding Volume Hierarchies

Approaches exist for certain specific bounding volume hierarchies for Rigid Objects. These can also be used in generating the bounding volumes for the nodes of articulated Rigid Body trees. Most existing approaches however, deal with homogeneous bounding volume trees, although there are useful studies on building a tree with different classes of SSVs [55] and C-trees[97]. It would be worthwhile to further investigate techniques for generating efficient bounding volume trees made up of fully heterogeneous nodes.

Furthermore, many of the existing bounding volumes generation techniques approach the problem with the intention of using the generated BVH trees as a broad phase culling technique, or for simple "yes/no" contact determination. As a result they tend to focus on reducing workload incurred in traversal of the trees *i.e.* reducing branching factors even at the expense of tightness of fit. Although the required result is not hugely different for a refinable contact modeller, it may be sometimes worthwhile to consider the specific needs of contact modelling when generating the hierarchies. For instance, an increase in the regularity of nodes and reduction in overlap between sibling nodes can be useful improvements for the approximate contact modeller, as can a pre-computed occupancy value for bounding volume nodes.

Appendix A

Related Publications

1. C. O'Sullivan, J. Cassell, H. Vilhjalmsson, **J. Dingliana**, S. Dobbyn, B. McNamee, C. Peters, and T. Giang [2002]. "Levels of Detail for Crowds and Groups", *Computer Graphics Forum*, 21(4), 2002.

2. C. O'Sullivan, **J. Dingliana** [2002]. "Gaze-contingent algorithms for Interactive Graphics", *The Mind's Eyes: Cognitive and Applied Aspects of Eye Movement Research. Hyönä, J. Radach, R. and Deubel, H. (Eds.).* Elsevier Science, Oxford. (To appear).

3. C. O'Sullivan, J. Cassell, H. Vilhjálmsson, S. Dobbyn, C. Peters, W. Leeson, T. Giang, and **J. Dingliana** [2002]. "Crowd and Group Simulation with Levels of Detail for Geometry, Motion and Conversational Behaviour". *Proceedings of Eurographics Ireland 2002*, pp 15-20.

4. C. O'Sullivan, and **J. Dingliana**, [2001]. "Collision Handling and Perception", *ACM Transactions on Graphics.* June 2001.

5. C. O'Sullivan, **J. Dingliana**, F. Ganovelli, and G. Bradshaw [2001], "Collision Handling for Virtual Environments", *Eurographics 2001, Tutorials Program, Manchester UK.*

6. C. O'Sullivan, **J. Dingliana**, G. Bradshaw, and A. McNamara [2001], "Eye-tracking for Interactive Computer Graphics", Presented at the 11th European Conference on Eye Movements (ECEM 11), Turku, Finland, 2001.

7. **J. Dingliana**, C. O'Sullivan, and G. Bradshaw [2001], "Collisions and Adaptive Levels of Detail", Presented at SIGGRAPH 2001 Sketches and Applications Program, LA 2001.

8. C. O'Sullivan, **J. Dingliana** [2001], "Real vs. Approximate Collisions: When can we tell the difference?", Presented at SIGGRAPH 2001 Sketches and Applications Program, LA 2001

9. **J. Dingliana**, and C. O'Sullivan [2001] "Levels of Detail in Physically-based Real-time Animation", *ERCIM News (special issue on Computer Graphics and Visualization)*, Issue 44 [2001], pp 43-44.

10. **J. Dingliana**, and C. O'Sullivan [2000], "Graceful Degradation of Collision Handling for Physically Based Animation", *Computer Graphics Forum* Vol.20 Num.3 (Eurographics 2000 Proceedings).

11. F. Ganovelli, **J. Dingliana**, and C. O'Sullivan [2000], "BucketTree: Improving collision detection between deformable objects", Proc. Spring Conference in Computer Graphics (SCCG2000), Bratislava, April 2000. pp 156-163.

12. C. O'Sullivan, and **J. Dingliana** [1999], "Collision Detection and Response Using Sphere Trees", Proc. Spring Conference in Computer Graphics.

Appendix B

Implementation Details

Some key data structures and functions are provided here to illustrate some of the concepts discussed in chapter 3.

B.1 Selected Data Structures

B.1.1 Sphere Tree

```
struct Stree {

    Stree **_children;
    Stree *_parent;

    double _radius;
    Vector _position;
    int _numChildren;
};
```

B.1.2 Sphere Hit

```
struct sphereHit
```

```
{
    Vector colDir;
    Vector colPos;

    Stree* tree1;
    Stree* tree2;

    RigidBody* thing1; //owner of tree1
    RigidBody* thing2; //owner of tree2

    bool isLeafCollision;
    sphereHit* nextNode; //may need this
};
```

B.1.3 Collision

```
struct Collision
{
    List<sphereHit> verHits; //verified processed
    Collision *nextNode;
};
```

B.2 Selected Pseudocode

B.2.1 Sphere-tree collision detection

The following pseudocode example illustrates how collision detection queries between two sphere tree data structures are performed recursively.

```
function collisionDetect(sphereTree1, sphereTree2)
{
```

```
apply worldcoordinate transform to sphereTree1._position
    and to sphereTree2._position

if colliding(sphereTree1, sphereTree2)
{
    if (sphereTree1.isLeafNode() && sphereTree2.isLeafNode())
        return true;
    else
    {
        if (time remains for further collision queries)
        {
          coll = false;
          for all children of sphereTree1
          {
            coll = coll || collisionDetect(sphereTree2,
                                              sphereTree1->child)
          }
          if coll //at least one leaf node collision
             return true
          else
             return false
        }
        else //no time left to check further
            return true //so assume they are colliding
    }

}
else
    return false;
```

```
}
```

B.2.2 Progressive refinement phase

In practice, a recursive function is not used, instead all possible collisions are stored in a collision list. Each collision is refined one at a time if time remains for further processing as in the following function.

```
function refineCollision(currentCollision)
{
    for all sphereHits in currentCollision
    {
        get currentSphereHit
        get tree1;
        get tree2;

        for all children of tree1
        {
            get tree1->child

            if colliding(tree1->_child, tree2)
            {
                insert sphereHit (tree2, tree1->child)
                    into sphereHit List
                sphereCollisions+=1;
            }
        }
    }
    if (sphereCollisions == 0)
    delete currentCollision from collision list
}
```

Bibliography

[1] W. W. Armstrong and M. W. Green. The dynamics of articulated rigid bodies for purposes of animation. In *Proc. Graphics Interface '85*, pages 407–415, 1985.

[2] H. Aubert and O. Förster. Beiträge zur kentniss des indirekten sehens (i): Untersuchungen über den raumsinn der retina. *Arch. Ophthamology*, 3:1–37, 1857.

[3] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(4):223–232, 1989. SIGGRAPH '89 Proceedings.

[4] David Baraff. *Dynamic Simulation of Non-Penetrating Rigid Bodies.* PhD thesis, Department of Computer Science, Cornell University, May 1992.

[5] R. Barzel and A. H. Barr. Modelling with dynamics constraints. ACM siggraph '87 Course Notes, 1987.

[6] Ronen Barzel, John F. Hughes, and Daniel N. Wood. Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation '96*, pages 183–197, 1996.

[7] G. R. Bradshaw. *Bounding Volume Hierarchies for Level of Detail Collision Handling.* PhD thesis, Computer Science Department, Trinity College Dublin, May 2002.

[8] G. R. Bradshaw and C. O'Sullivan. Sphere-tree construction using medial-axis approximation. In *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation SCA 2002*, 2002.

[9] S.A. Camereon. Collision detection by four-dimensional intersection testing. *IEEE Transactions on Robotics and Automation*, 6(3):291–302, 1990.

[10] S. A. Cameron and R. K. Culley. A study of the clash detection problem in robotics. In *Proceedings of the IEEE ICRA*, pages 591–596, 1986.

[11] S.A. Cameron. Enhancing gjk: Computing minimum penetration distances between convex polyhedra. In *Proceedings of the Int. Conf. On Robotics and Automation.*, pages 3112–3117, 1997.

[12] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge MA, 1988.

[13] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Graphics Interface '97*, pages 1–8. Canadian Human-Computer Communications Society, 1997.

[14] S. Chenney and D. Forsyth. View-dependent culling of dynamic systems in virtual environments. In *ACM Symposium on Interactive 3D Graphics*, pages 55–58, 1997.

[15] S. Chenney, J. Ichnowski, and D.A. Forsyth. Dynamics modeling and culling. *IEEE Computer Graphics and Applications*, 19(2):79–87, 1999.

[16] Stephen Chenney. Simulation level-of-detail. In *Proceedings of Game Developers Conference 2001*, 2001.

[17] Stephen Chenney. Scalable dynamics. Talk in Siggraph 2002 Course Notes 35: Super Sizeit! Scaling up to Massive Virtual Worlds, 2002.

[18] Stephen Chenney, Okan Arikan, and D. A. Forsyth. Proxy simulations for effcient dynamics. In *Eurographics 2001 Short Presentations*, 2001.

[19] Stephen J. Chenney. *Controllable and Scalable Simulation for Animation.* PhD thesis, University of California, Berkeley, 2000.

[20] J. Clement. Students' preconceptions in introductory mechanics. *American Journal of Physics*, 50(1):66–71, 1982.

[21] J.D. Cohen, M.C.Lin, D.Manocha, and M.K.Ponamgi. I-collide: An interactive and exact collision detection system for large-scaled environments. In *Proceedings of ACM Int.3D Graphics Conference*, pages 189–196, 1995.

[22] S. Collins. Computer graphics during the 8-bit computer game era. *Computer Graphics*, 32(2), May 1998.

[23] G. Van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[24] G. Van den Bergen. A fast and robust gjk implementation for collision detection of convex objects. Tech Report, Department of Mathematics and Computer Science Eindhoven University of Technology, 1999.

[25] Edward A. Desloge. *Classical Mechanics, Volume 1.* Wiley, Interscience, 1982.

[26] John Dingliana and Carol O'Sullivan. Graceful degradation of collision handling in physically based animation. *Computer Graphics Forum (Eurographics 2000 Proceedings)*, 19(3):239–247, 2000.

[27] M.A. Duchaineau, M. Wolinsky, D.E. Sigeti, M.C. Miller, C. Aldrich, and M.B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. In *Proceedings of IEEE Visualization '97*, pages 81–88, 1997.

[28] Stephan A. Ehmann and Ming C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 500–510. Blackwell Publishing, 2001.

[29] Routh E.J. *Dynamics of a system of Rigid Bodies.* Macmillan and Company, Ltd., London, 1905.

[30] T.A. Funkhouser and C.H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings SIGGRAPH '93*, pages 247–254, 1993.

[31] F. Ganovelli, J. Dingliana, and C. O'Sullivan. Buckettree: Improving collision detection between deformable objects. Proceedings of the Spring Conference on Computer Graphics (SCCG), 2000.

[32] T. Giang, G. Bradshaw, and C. O'Sullivan. Complementarity based multiple point collision resolution. Trinity College Dublin, Computer Science Technical Reports, 2002.

[33] E. G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Trans. Robotics and Automation*, 4(2):193–203, 1988.

[34] D. Gilden and D. Proffitt. Understanding collision dynamics. *Journal of Experimental Psychology: Human Perception and Performance.*, 15(2):372–383, 1989.

[35] W. Goldsmith. *Impact: The theory and Physical Behaviour of Colliding Solids.* Edward Arnold Pub. Ltd., London, 1960.

[36] Herbert Goldstein. *Classical Mechanics.* Addison Wesley, Reading MA, 1950.

[37] S. Gottschalk, M.C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proceedings SIGGRAPH '96*, pages 171–180, 1996.

[38] James K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988. SIGGRAPH 88 Proceedings.

[39] D. Haumann. Modelling the behaviour of flexible objects. ACM siggraph '87 course notes, 1987.

[40] Taosong He and A. Kaufmann. Collision detection for volumetric objects. In *Proceedings of the 8th IEEE Visualization '97 Conference*, pages 27–734, October 1997.

[41] J.K. Hodgins, J.F. O'Brien, and J. Tumblin. Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics.*, 4(4):307–316, 1998.

[42] K. E. Hoff III, A. Zaferakis, M. Lin, and D. Manocha. Fast 3d geometric proximity queries between rigid and deformable models using graphics hardware acceleration. UNC-CS Technical Report, University of Nort Carolina, Chapel Hill, 2002., 2002.

[43] P. Hubbard. Real-time collision detection and time-critical computing. In *Proceedings of the 1st Workshop on Simulation and Interaction in Virtual Environments, U. of Iowa, July 1995.*, pages 92–96, 1995.

[44] Philip M. Hubbard. Space-time bounds for collision detection. Technical Report CS-93-04, Department of Computer Science, Brown University, 1993.

[45] Philip M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, 1995.

[46] Phillip M. Hubbard. Interactive collision detection. In *IEEE Symposium on Research Frontiers in VR 1993, San Jose, California*, pages 24–32, October 1993.

[47] Phillip M. Hubbard. *Collision Detection for Interactive Graphics Applications.* PhD thesis, Department of Computer Science, Brown University, April 1995.

[48] P.M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[49] M. K. Kaiser and D. R. Proffitt. Observers' sensitivities to dynamic anomalies in collisions. *Perception and Psychophysics*, 42(3):275 – 280, 1987.

[50] T. Kane and D. Levinson. *Dynamics: Theory and Applications.* McGraw-Hill, New York, N.Y., 1985.

[51] Y. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. In *Proceedings of the ACM SIGGRAPH Symposium on Computer Architecture 2002*, pages 23–31, 2002.

[52] Young J. Kim, Miguel A. Otaduy, Ming C. Lin, and Dinesh Manocha. Fast penetration depth computation for physically-based animation. In *ACM Symposium on Computer Animation*, July 2002.

[53] Y. Kitamura, H. Takemura, N. Ahuja, and F. Kishino. Efficient collision detection among objects in arbitrary motion using multiple shape representations. In *Proceedings 12th IAPR Int. Conf. On Pattern Recognition*, volume 1, pages 390–396, 1994.

[54] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[55] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with sphere swept volumes. UNC Chapel Hill Computer Science Technical Report TR99-018, 1999.

[56] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast distance queries using rectangular swept sphere volumes. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2000.

[57] A. Leslie and S. Keeble. Do six-month-old infants perceive causality? *Cognition*, 25:265–288, 1987.

[58] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[59] M. Lin. *Efficient Collision Detection for Animation and Robotics.* PhD thesis, University of California, Berkeley, 1993.

[60] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *Proc. IEEE Conference on Robotics and Automation*, pages 1008–1014, 1991.

[61] L. Loschky and G. McConkie. User performance with gaze contingent multiresolution displays. In *Proceedings of the Eye Tracking Research and Applications Symposium*, pages 97–103, 2000.

[62] D. Luebke, B. Hallen, D. Newfield, and B. Watson. Perceptually driven simplification using gaze-directed rendering. Technical report, CS-2000-04 University of Virginia, 2000.

[63] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. In *Proc. ACM SIGGRAPH'92 Conference*, pages 309–312, July 1992.

[64] A. Michotte. *The Perception of Causality.* Basic Books., New York, 1963.

[65] B. Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics.*, 17(3):177–208, 1998.

[66] B. Mirtich. Timewarp rigid body simulation. In *Proceedings SIGGRAPH 2000*, pages 193–200, 2000.

[67] Brian Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems.* PhD thesis, University of California, Berkeley, 1996.

[68] Brian Mirtich and John F. Canny. Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, pages 181–188, 217, 1995.

[69] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics (SIGGRAPH 1988, Proceedings)*, 22:289–298, 1988.

[70] D. Obrien, S. Fisher, and M. Lin. Automatic simplification of particle system dynamics. In *Proceedings of Computer Animation 2001*, pages 210–218, 2002.

[71] T. Ohshima, H. Yamamoto, and H. Tamura. Gaze-directed adaptive rendering for interacting with virtual space. In *Proceedings IEEE VRAIS'96*, pages 103–110, 1996.

[72] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees. In *Proceedings of the Spring Conference in Computer Graphics, Bratislava*, pages 83–92, 1999.

[73] C. O'Sullivan and J. Dingliana. Collisions and perception. *ACM Transactions on Graphics*, 20(3), July 2001.

[74] C. Osullivan, R. Radach, and S. Collins. A model of collision perception for real-time animation. *Eurographics Workshop on Computer Animation and Simulation*, pages 67–76, 1999.

[75] Carol O'Sullivan. *Perceptually-Adaptive Collision Detection for Real-time Computer Animation.* PhD thesis, University of Dublin, 1999.

[76] I.J. Palmer and R.L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, 1995.

[77] M.K. Ponamgi, D. Manocha, and M.C.Lin. Incremental algorithms for collision detection between polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):51–64, 1997.

[78] D. Profitt and D. Gilden. Understanding natural dynamics. *Journal of Experimental Psychology: Human Perception and Performance.*, 15(2):384–393, 1989.

[79] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Proceedings of Graphics Interface '97*, pages 177–189, 1997.

[80] W.E. Red. Minimum distances for robot task simulation. *Robotics*, 1:231–238, 1983.

[81] M. Reddy. *Perceptually Modulated Level of Detail for Virtual Environments.* PhD thesis, University of Edinburgh, 1997.

[82] M. Reddy. Perceptually optimized 3d graphics. *IEEE Computer Graphics and Applications*, 21(5):68–75, 2001. Sept/Oct.

[83] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. In *Computer Graphics (SIGGRAPH 97 Conference Proceedings)*, pages 345–352. ACM SIGGRAPH, 1997.

[84] J. Saarinen. Visual search for global and local stimulus features. *Perception*, 23:237–243, 1994.

[85] H. Strasburger, I. Rentschler, and L. O. Jr. Harvey. Cortical magnification theory fails to predict visual recognition. *European Journal of Neuroscience*, 6:1583–1588, 1994.

[86] K. R. Symon. *Mechanics.* Addison Wesley Publishing Compnany, Reading MA, 1971.

[87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, July 1987.

[88] R.B.H. Tootell, M.S. Silverman, E. Switkes, and R.L. De Valois. Deoxyglucose analysis of retinotopic organization in primate striate cortex. *Science*, 218:902–904, 1982.

[89] A. Treisman. Perceptual grouping and attention in visual search for features and for objects. *Journal of Experimental Psychology: Human Perception and Performance*, 8:194–214, 1982.

[90] P. Volino and N. Magnenat-Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum (EuroGraphics Proceedings)*, 13(3):155–166, 1994.

[91] B. Watson, N. Walker, L.F. Hodges, and A. Worden. Managing level of detail through peripheral degradation: effects on search performance in a head mounted display. *ACM Trans. Computer-Human Interaction*, 4(4):323–346, 1997.

[92] B.A. Watson, A. Friedman, and A. McGaffey. Measuring and predicting visual fidelity. In *Proceedings SIGGRAPH 2001*, pages 213–220, 2001.

[93] J. Weil. The synthesis of cloth objects. *Computer Graphics*, 20(4):49–54, August 1986.

[94] R.W. Weymouth. Visual sensory units and the minimal angle of resolution. *American Journal of Ophthamology*, 46:102–113, 1958.

[95] J. Wilhelms and B. A. Barsky. Using dynamic analysis for the animation of articulated bodies such as humans and robots. In *Proc. Graphics Interface '85*, pages 97–104, May 1985.

[96] Andrew Witkin, David Baraff, and Michael Kass. Physically based modelling. In *Siggraph 2001 Course Notes 25.*, 2001.

[97] J.H. Youn and K. Wohn. Realtime collision detection for virtual reality applications. In *Proceedings IEEE Virtual Reality Annual Int. Sym.*, pages 18–22, 1993.