1-1-2004

Network-based vehicle collision detection and simulation

Liangshou Wu
Iowa State University

Network-based vehicle collision detection and simulation

by

Liangshou Wu

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
James Oliver, Major Professor
Adrian Sannier
Carolina Cruz-Neira

Iowa State University

Ames, Iowa

2004

Graduate College

Iowa State University

This is to certify that the master's thesis of

Liangshou Wu

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My foremost thank goes to my wife Ruiju Huang. Without her help and support, I would have not finished this thesis. She helps me a lot on both my study and personal life. Her full heart dedication on taking care of our little baby makes me be able to concentrate on finishing this thesis and preparing my defense. Special thanks go to my committee members, specifically Dr. Jim Oliver and Dr. Adrian Sannier for their advice and guidance on system architecture design and many specific implementation details. I highly appreciate them for giving me the great opportunity to work and study in VRAC. Thanks to Dr. Jim Oliver and Dr. Adrian Sannier again for their patience on revising my thesis and relentless editorial corrections. I would like to thank Dr. James Bernard for his knowledge of vehicle collision dynamics. Thanks also to Bryan Walter for giving me highlight on networking programming and Ole Balling for his kindness to guide me into VDANL. Thanks to Mark Knight for helping me to start this thesis at the very beginning. I also want to thank Karen Peretti for modifying the scene model. Finally, I would like to express my gratitude to my family for their lifelong support.

ABSTRACT

Vehicle driving simulation, collision detection, and collision simulation of rigid bodies are not new in their corresponding literature, but the integration of all these techniques is a challenging and interesting topic. Some special requirements arise when they are combined, especially when multiple vehicles, located at different places on a network, are involved in collision simulation.

This thesis implements a network-based vehicle collision detection and response simulation system. This system has all the components that are required by vehicle driving simulation. It supports vehicle-to-scene and vehicle-to-vehicle collision detection and response simulation in real-time required by human-in-the-loop driving simulation. Additionally, it supports collaborative driving simulation for multiple vehicles in the same virtual environment operated from different physical locations. It provides consistent and realistic collision response for vehicles that collide.

A network-based collision server is developed to accommodate vehicle-to-vehicle collision detection and response simulation. A general collision algorithm supplies consistent collision result for all collided vehicles. The method takes advantage of Open Scene Graph's (OSG) built-in collision detection functionality for vehicle-to-scene collision detection. For vehicle-vehicle collision detection a two stage process is introduced which employs a bounding circle localization technique and Cohen-Sutherland clipping. A simple but realistic vehicle-to-vehicle and vehicle-to-scene, collision simulation algorithm is developed with a friction model and a modified coefficient of restitution model, based on a vehicle collision simulation algorithm presented by Macmillan.

Implementation testing shows that the network-based collision simulation system can provide a real-time, realistic and robust system in a network with relatively small time lags delay, such as a LAN, a city network, and some inter-city networks. The implementation has

demonstrated support for simultaneous collision simulation with up to 32 vehicles operating at reasonable speed in local area network.

1. INTRODUCTION

1.1. Motivation

In companies, universities, and research institutes around the world, there are at least one hundred vehicle simulators in use with different levels of ability [1]. Usually, they are used for vehicle design evaluation, driver behavior studies, or traffic safety studies. Real-time human-in-the-loop simulators can give the driver a very "real" feeling by introducing stereo visual, audio, force feedback, and physical motion effects. Vehicle simulators generally do not incorporate collision simulation.

Like most engineering simulations, methods for representing the collision of moving bodies ranges from the highly accurate and hence computationally demanding techniques, such as non-linear finite element impact analyses, to the physically simple and computationally less demanding, such as a ball bouncing off a wall in a computer game. Physically accurate vehicle collision or crash evaluation is done offline due to its high computational requirements. Detailed non-real-time vehicle collision simulations have been used for more than forty years in the automobile industry for assessing a vehicle's collision safety, and in traffic accident cases for reconstructing accidents.

Meanwhile, with continuing rapid development of computer technology, virtual reality (VR) is now very popular. For many VR applications, simple collision detection is used solely for user's navigation. More complex collision simulation is used in relatively few of them, generally those designed for certain special purposes, such as virtual prototyping and assembly simulation. Many 3D games in the market involve some sort of vehicular interaction (racing, etc.). These generally have some specialized vehicle collision response simulation, but most of them are simplified to maintain real-time performance. Sometimes, the collision response is not reasonable and not consistent for multi-user games, especially for those designed to operate in a distributed networked environment..

In this thesis, a vehicle-to-vehicle collision detection and simulation algorithm is developed and implemented in addition to vehicle-to-scene collision simulation such as in Knight's thesis [2]. The collision algorithm is a compromise between a non-real-time detailed collision simulation algorithm and very simple, but often physically unreasonable, vehicle collision simulation algorithm. The result is a real-time, physically reasonable, and realistic collision algorithm. Additionally, it supports the consistent collision responses for multi-vehicle simulation by introducing a network-based collision server, called the collider server. It eliminates or minimizes the inconsistency of collision results caused in multi-vehicle simulation in which the collision algorithm resides on clients running individual vehicle simulations as well. A new networking architecture is developed to realize the functionalities of the collider. The integration and implementation of different simulation components (vehicle dynamics model, image generator, network server, and collision algorithm) is described as well. The resulting system has all the basic components needed to support a realistic multi-user virtual driving simulator.

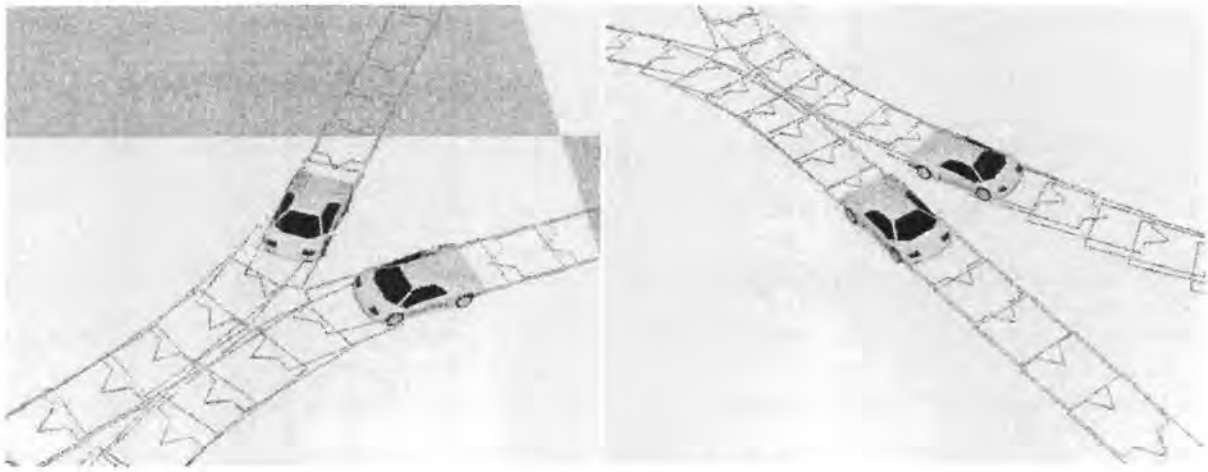

Figure 1-1 Side by Side Collision Simulation

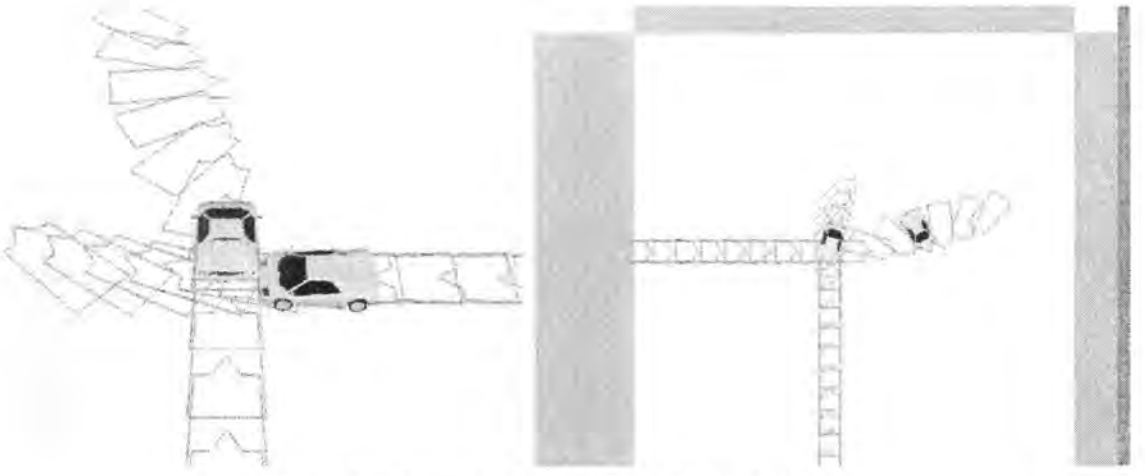

Figure 1-2 Front-to-Side Collision Simulation

1.2. Thesis Organization

Chapter 2 gives a brief literature review of vehicle driving simulation, collision detection, and rigid body collision simulation (vehicle-to-scene collision simulation), and presents the specific methods and algorithms used in this thesis. Chapter 3 through chapter 6 explains the details of the network-based vehicle collision simulation system. Chapter 7 presents test results of the usability of the whole simulation system. Chapter 8 gives the conclusion for this thesis, and points out remaining challenges and suggestions for future works.

2. LITERATURE REVIEW

2.1. Driving Simulation

Driving simulation technology stems from its predecessor, aircraft simulation. The earliest flight simulator started in 1910. In early 1970s, General Motors and Virginia Polytechnic Institute and State University created the first human-in-the-loop driving simulator that included a 16 degree of freedom (DOF) vehicle dynamics model and a small motion base [1]. Since that time, more than one hundred driving simulators have been developed around the world from very simple to very advanced. The simple ones generally have a fixed base with a simple vehicle dynamics model and relatively poor graphics effects. The very advanced ones, like the National Advanced Driving Simulator (NADS) at the University of Iowa, generally have a very powerful and precise motion base, high fidelity vehicle dynamics model, high quality visual and audio generation system, and so on. No matter how simple or advanced a driving simulator is, it includes four basic components [1,3,4]:

- Real-time simulation of a complex physical system.
- Simulation of environment.
- Control devices.
- Visual and audio system.

A driving simulator gives a user the impression of actual driving by simulating and visualizing the environment changes, sound changes, hepatic force feedback and motion changes [5,6,7]. With continually increasing network bandwidth, more and more recent research is focused on joint or collaborative driving simulations distributed on network that support multi-driver rather than single driver operation. Balling, et al. [8] is an example of collaborative driving simulator that supports dual-location driving simulation. Based on the collaborative driving simulation, Knight moved further by adding vehicle-to-scene collision

simulation [2]. The collision simulation algorithm in his thesis works as an external module of a commercially available vehicle dynamics model called VDANL [25]

2.2. Collision Detection and Response Simulation

Collision detection has been extensively pursued in the fields of robotics and computational geometry. For different collision models, different collision detection methods are used. Lin and his colleagues presented various collision detection algorithms in [9, 10, 11, 12]. Most of them can be used for complicated and large-scaled collision detection scenarios. But they are unsuitable for real-time simulation, especially for situations where objects move quickly.

In the fields of physical-based simulation, computer graphics, and computer animation, collision detection and response animation is not new. For most cases, the simulation and animation require real-time, so more and more simplified and limited collision detection algorithms are developed. The typical simplified collision detection method is collision detection for convex polygons [13]. This method can be used in several situations, such as user navigation, by representing the user as a box, ray picking, and simple object collision detection (such as box, sphere, and cylinder).

In computer graphics and animation, there are numerous algorithms for dynamics simulation of rigid bodies [14, 15, 16, 17]. But most of them pay more attention to pure static or pure dynamic simulation than collision simulation. For collision simulation, Moore and Wilhelms presented a method to simulate the collision force. They represent the collision force between two penetrating objects with springs and there is no friction force involved in their collision response. Kawachi, Suzuki and Kimura used impulse friction force to simulate rigid body motion and collision response. They used static and dynamic friction impulse to simulate the friction force. But the static impulse is hard to implement in practice. The coefficient of restitution method is used for normal force calculation.

Kamal presented a method for computer analysis and simulation of vehicle to barrier impact [18]. The method used is valid for only some typical collision cases like car crashworthiness tests. Greene showed a method for computer simulation of car-to-car collisions [19]. His goal is to simplify the simulation process of car-to-car collision and provide a comparison of simulation results with crash test data. Both methods are relatively complex, and hence more accurate.

2.3. Research Emphasis

The research presented in this thesis addresses the needs of multi-user network distributed driving simulation. The primary challenge is to provide collision detection and realistic post-collision dynamics of vehicles at rates that enable real-time human-in-the-loop interaction.

For vehicle-to-scene collision detection this work takes advantage of the ray-surface intersection functionality of an open source scene graph rendering package called Open Scene Graph [20]. A bounding circle and the Cohen-Sutherland clipping algorithm [21] are implemented for vehicle-to-vehicle collision detection. Macmillan [22] presents a simple collision response calculation algorithm for vehicle-to-vehicle, and vehicle-to-scene, collision simulation. This method represents the vehicles as simple rectangles, and treats the collision process as a singular atomic step through out which the collision force is constant, and the vehicle states remain unchanged. In this research, a new vehicle-to-vehicle collision algorithm is developed based on Macmillan's algorithm. The new algorithm is improved to be provided more realistic and general behavior by incorporating a friction force model and extending the coefficient of restitution model.

Walter [23] presents a basic networking architecture for multi-vehicle simulation. This research improves on Walter's approach for dealing with vehicle-to-vehicle collision detection

and simulation and safe quitting. A new collision server is added to realize consistent collision response for collided vehicles in the same detected collision.

An independent and robust image generator (IG), that is able to run in desktop mode and immersive mode, is developed by using Open Scene Graph and VR Juggler [24] for multi-vehicle driving simulation. It has the basic functionality required by any VR application and necessary for multi-vehicle driving simulation.

VDANL is a commercial vehicle dynamics simulator that has 17 degree of freedom (DOF) [25]. A new Visual Basic (VB) interface is developed to integrate the IG, networking API and VDANL. Also, a new VDANL UDM dll is developed for applying collision response from the collision algorithm to VDANL integration loops. Additionally, a simplified vehicle dynamics model is implemented according to Gillespie's vehicle dynamics theory [26]. The simplified VDM is extremely useful for testing when many versions of commercial VDM is not available or the frame rate of commercial VDM is not sufficient.

3. NETWORK ARCHITECTURE AND SCHEME

As well-designed software architecture is important for robust software development. A well designed architecture not only enhances maintenance and further development, but is also critically important for the effectiveness, robustness and correctness of the resulting application. In this chapter, the network architecture designed for vehicle-to-vehicle collision detection is presented. The communication scheme of each specific process between each component of the architecture will also be described. Finally, this chapter presents the pros and cons of this architecture.

3.1. Network Architecture

Many general network protocols, like TCP/IP, operate in the client-server style. The network architecture used in this research has the same style. As shown in Figure 3-1, the architecture consists of two main components, the surrounding clients and the central collider server. The common feature of client-server style architecture is that clients connect to the server, and the server serves clients. The differences are how the server servers clients, and what the clients do.

Basically, the collider server used in this research has two subcomponents, the network server and the collision server. The network server behaves as a connector. Each client connects to the server and sends a message to it. The server passes the message from every client to all the others. The network server manages the joining and quitting of clients, while it keeps other clients running normally. The collision server does collision detection and response calculation, and sends out collision message to corresponding clients via the network server. The collision can be vehicle-to-scene collision or vehicle-to-vehicle collision.

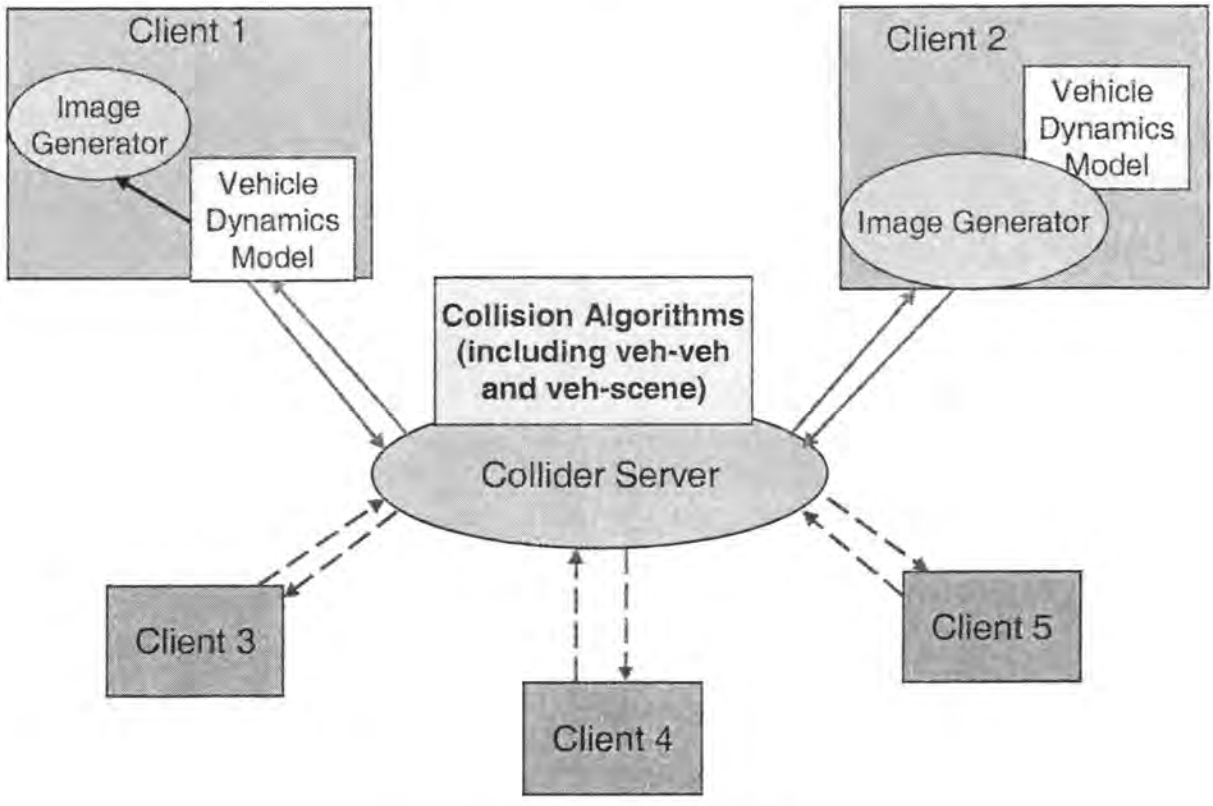

Figure 3-1 Network Architecture

On the client side, there are two subcomponents; the image generator (IG) and the vehicle dynamics model (VDM). As shown in Figure 3-1, image generator may connect to the collider server directly or connect to the vehicle dynamics model and then connect to the collider server. Both methods are useful for different conditions. Usually, the graphics frame rate of IG is low down to 20-60 Hz, it is acceptable for computer graphics. But for dynamics integration, too large integration step decreases the accuracy of dynamics results. Additionally, too large frame time causes failed collision detection that will be explained in chapter 7. So generally, it is better to separate IG and VDM to put them on different machine. The separation also makes VDM be capable of doing more complex dynamics calculation. If IG and VDM are simple enough, it is good to combine them together. That makes communication simpler. For this research, VDM is separated from IG.

3.1.1. Network Server

Network server, like a bridge or connector, connects all the clients together, receiving and forwarding incoming messages. It is also a bridge between the collider and all the clients. While the server forwards the messages to other clients, it can also forward the messages to collider. The server also takes care of the proper initialization and safe ending of each client.

Additionally the network server supplies a consistent timer for both server and clients so that the simulation is based on the same timing. The last functionality of network server is to synchronize the server and clients. Synchronization is important because it avoids flooding the server or other clients, or starving the server. Synchronization ensures that the sending rate of packets from each client is such that both the server and clients have enough time to process the packets, while simultaneously keeping the sending rate as high as possible to optimize the collision and dynamics accuracy, and visualization performance.

3.1.2. Collider

Although the collider is part of the collider server's functionality, it is actually an independent module that can be used with either server side code or client side code. Basically, the collider is a collision center. It gets vehicles' states from clients or the network server, and does the collision detection for the vehicle-to-scene and vehicle-to-vehicle conditions. If a collision is detected, the responses for the corresponding client (or clients) will be calculated and passed to the network server and then back to that client (or clients). The specific collision detection and response calculation mechanism and algorithms will be explained in detail in Chapters 4 and 5.

Besides collision detection and simulation, the collider also needs to manage the list of simulated vehicles, the environment scene model, the vehicle collision model, and avoid duplicating the same collision.

3.1.3. Image Generator (IG)

The IG is a visualization program that depicts vehicle driving and collision simulation in a virtual environment in real time. In this research, it is a client that connects to the collider server directly or indirectly. It receives new packets from the vehicle dynamics model or other clients, and updates all vehicles' state and environment variables. As a visualization program, it has the typical basic functionality, such as navigation, camera/view specification, model loading, and environment simulation.

In order to improve graphics effects, it is often necessary to predict the vehicles' states when there is no new information available for a vehicle or the IG is running faster than the packet update rate. To make the vehicles move smoothly in these situations, the dead-reckoning technique is implemented. The image generator must also deal with the situation in which multiple vehicles are involved the simulation. It needs to add or remove vehicles from the 3D scene whenever the new client joins or quits the simulation. Since the IG does not connect to the server directly it sometimes difficult to notify it of a client quitting the simulation. The IG gets update packets only from the VDM, there is no direct information that indicates a client has quit. Of course, there should be no more new packets from a client if it has quit. But it is also possible that a communication problem may cause packet delay for certain clients. So the IG has to distinguish those two situations.

3.1.4. Vehicle Dynamics Model (VDM)

The vehicle dynamics model (VDM) does the integration calculation based on user inputs (steering angle, acceleration pedal, and brake pedal) and external force inputs (brake force, lateral force, and collision force). The outputs are the vehicle state values, including position, orientation, linear and angular velocity, and linear and angular acceleration. The vehicle dynamics model is an important component in vehicle driving simulation. Thus, it is also critical for vehicle collision simulation. Generally, the more accurate the dynamics model, the more accurate the collision simulation result will be.

However, the computational cost for a highly accurate VDM is can quickly exceed the requirements for real time simulation. In this research, both a highly accurate vehicle dynamics model and simplified vehicle dynamics model are used. The highly accurate vehicle dynamics model is the commercial available Vehicle Dynamics Analysis Nonlinear (VDANL) from System Technology, Inc. (STI), which has 17 DOF. For typical simulations on common PC platforms, the integration time step can reach 0.005 second (200fps) for VDANL. This is small enough for real time simulation in most cases, such as those with no extremely high vehicle speed, no large network delays. The simplified model used in this thesis has only 7 DOF and is created by author. Its integration time can be ignored compared to the 3D scene rendering time and collision detection time. Using the simplified VDM is quite flexible compared to VDANL. Either velocities or forces from the collision algorithm can be applied to the simplified VDM directly. The details about how the vehicle dynamics models are implemented are covered in chapter 6.

3.2. Communication Scheme

3.2.1. Launching Collider Server

The collider server must be launched before any client tries to connect to it. Although the collision algorithms can be running with either server side code or client side code, in this research, it is implemented on the server side for simplicity. When the collider server is up, the collision algorithms are also running as part of server. The server is launched with two inputs: one is the server port number, the other is the environment scene model that is used for vehicle-to-scene collision detection. Once the server is up, it is ready to handle the client connections.

As shown in Figure 3-2, the server checks the socket every running loop to serve any connection requests from clients. It also executes the collision algorithms if any vehicle is connected.

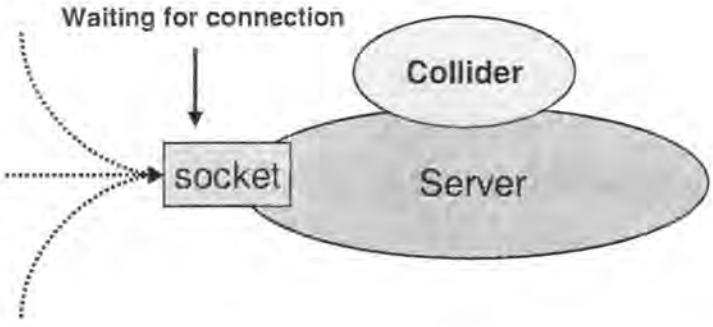

Figure 3-2 Collider Server

3.2.2. Launching IG and VDM

As described above, the IG and VDM can work independently or together. Their specific interaction is determined by the network architecture relationship among these two and the collider server. A distinct launching process is required for both the separated and combined cases, as described below.

- **Separated IG and VDM**

If the IG and VDM work independently, then the IG is just a rendering program that visualizes the result of the vehicle dynamics model driven by the user. Also, it provides visual results from other clients. The details about this will be shown in following sections. The VDM is the source of data relating to the IG and collider server. With a separate IG and VDM, the IG has two running threads: one for driving simulation, and the other for processing vehicle update packets. As shown in Figure 3-3, the IG server is one of two threads that accept the VDM's connection to get vehicle update packets from the VDM. So the IG (SERVER?) should be launched before the VDM starts. The IG server passes update packets to the image generator by shared memory. Information needed for running the IG includes the environment scene model that should be the same as the one used in the collider server.

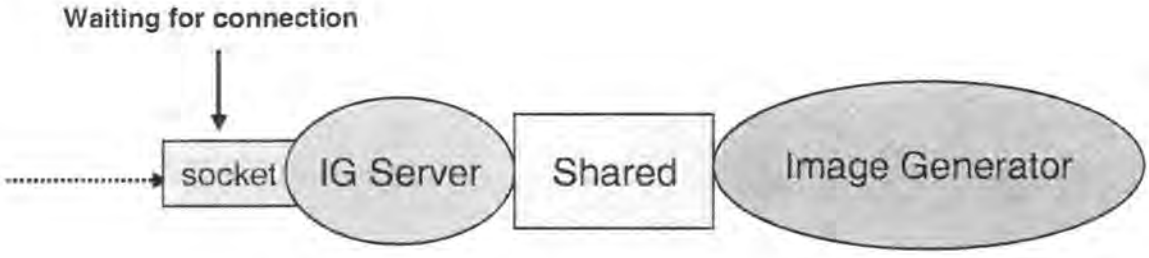

Figure 3-3 Image Generator

The VDM is must send data to both the IG and collider servers. Different VDMs may require different inputs. The arguments for running a VDM depend on which VDM is used. At least the VDM needs information about the IG server, collider server, and vehicle. Additional information may also include terrain data.

- **Bound IG and VDM**

If the IG and VDM are combined together, then there is no networking involved between the IG and VDM, and it is simpler for the application developer to interact with the IG and VDM. Running them requires information about the collider server, the environment scene model, and vehicle.

3.2.3. Opening Connection

The first thing that each client needs to do is to tell sever that a new client is going to join in. The collider server always accepts the connection from a client with a temporary additional channel. This temporary additional channel will become a permanent connection if the number of connections is still under the limit set by the application implementation. Based on the success or failure of the connection, different information will be communicated between clients and the collider server. As shown in Figure 3-4, the communication sequences between the collider server and clients are:

a) Client connects to server;
b) Collider server assigns a temporary channel to the client, and checks if the connection is allowed;

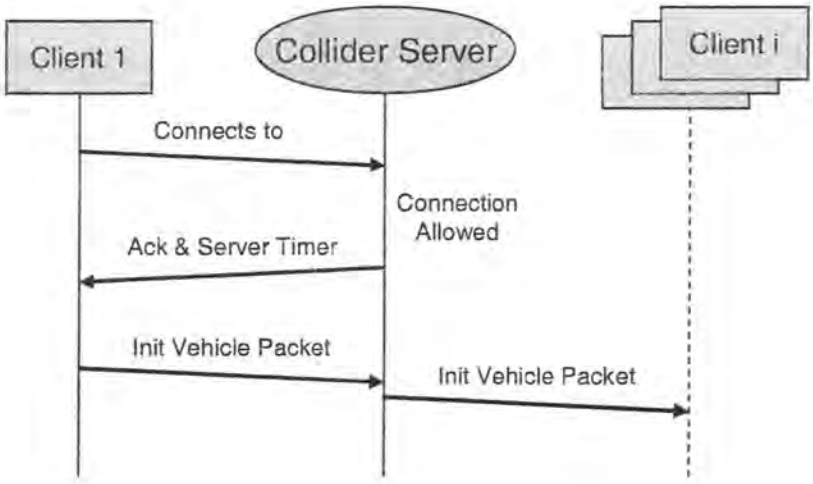

Figure 3-4 Successful Connection

If the connection is allowed, goes to c).

c) Server sends acknowledge packet and synchronization timer back to client;
d) Client sends initialization vehicle packet to collider server;
e) Collider server forwards the new client information to other clients and collision algorithms, then connection is done.

Otherwise as shown in Figure 3-5, it goes to f)

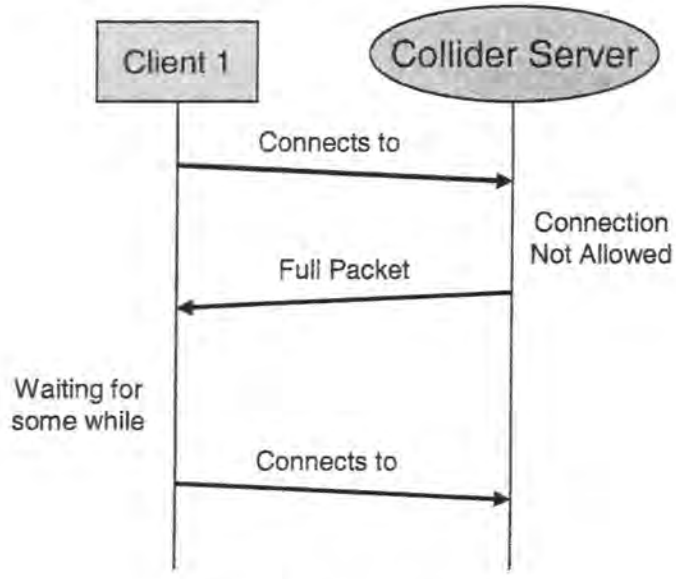

Figure 3-5 Failed Connection

f) Server sends connection full message back to client, and closes the connection;

g) Client keeps trying until time out or successful connection.

If the IG and VDM work independently and the client (or VDM) connects to the collider server successfully, then the client (or VDM) will try to connect to the IG. The process is very simple. The sequences are shown as Figure 3-6.

h) Client opens connection to IG;

i) IG sends acknowledgment packet back to client;

j) Client sends vehicle initialization packet to IG.

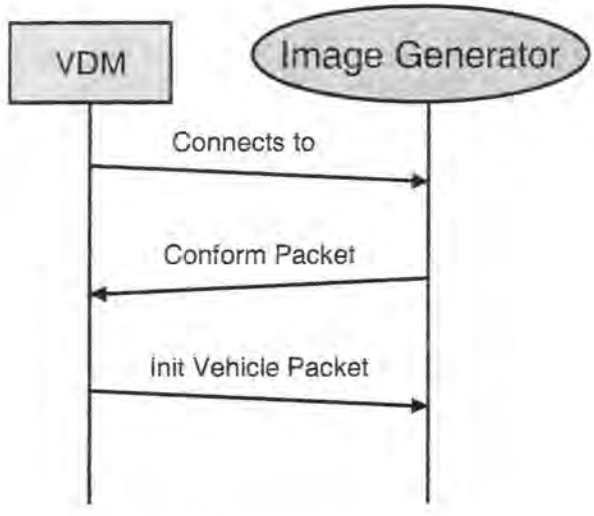

Figure 3-6 Connecting to IG

3.2.4. Regular Communication

Most of time during simulation, the primary information being passed is the vehicle update packet sent out by clients. Figure 3-7 shows the general state of the packet transmission process. Suppose that only one client sends out a packet at any given time. The client sends the packet to the collider server and the IG. If the IG and VDM are combined together, the VDM needs to send the same information to the IG, but it may not be in the same format. After the collider server gets the packet, it forwards the packet to the collision algorithms and all other clients except the one that sent the update packet. Then the collision

algorithms will do collision detection for the vehicle that has the same id as the update packet. All the clients (or VDMs) that get the update packet from the collider server forward the packet to the client IGs so that each IG can update the vehicle state. In real time simulation, all the clients send out update packets frequently. For the collider server, it just repeats the process described above. The result is that each IG connecting to the VDM gets update packets from all the clients frequently.

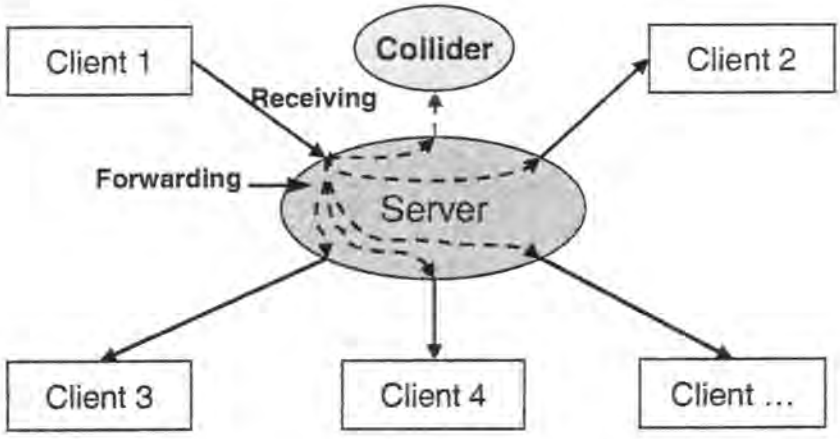

Figure 3-7 Update Packet Transmission

3.2.5. Collision Conditions

We didn't talk about what happens if collision algorithm detects the collision while doing collision detection. This section is about this topic. The collision happens when one vehicle hits the scene or one vehicle hits another vehicle if there are multiple vehicles available. For the first case, only one vehicle will get collision packet. For the second one, two vehicles will get the collision packets resulted from the same collision. Basically, the collision packet consists of collision response value, such as forces, moments, yaw rate, velocities.

Once VDM gets the collision packet, the collision forces and moments or the yaw rate and velocities would be the inputs for the next integration loop. So the vehicle's response

would be changed immediately after the next loop. And the new updating states will be sent to the IG connecting to this VDM and the collider server.

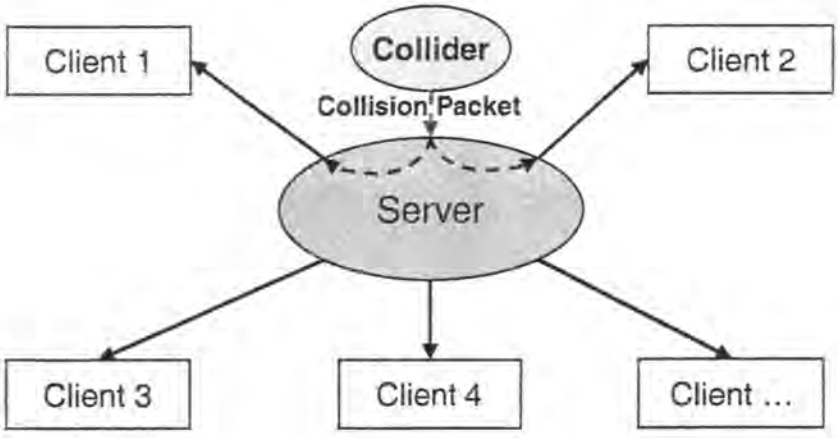

Figure 3-8 Vehicle-to-vehicle Collision

3.2.6. Closing Connections

The system should be robust with respect to one or more clients leaving the simulation at any given time. In other words, the overall simulation should continue correctly for all other clients. To ensure smooth transition, a client is permitted to quit only after getting permission from the collider server. The collider server then notifies the collision algorithm and all the other clients about of the departing client. This procedure is shown as Figure 3-9.

When a client gets a quit signal, it sends out the closing connection request to the collider server and waits for confirmation. The collider server forwards the closing request to the other clients and the collider. Once other clients and the collider get the closing message from that client, they remove that client from their vehicle list and scene. At the same time, the collider server sends a confirmation packet back to that client. After client gets the confirmation packet, it sends the closing packet to the IG, and quits immediately without waiting for confirmation from the IG. The IG removes all vehicles from the scene, clears its vehicle list, and waits for new connection from VDM.

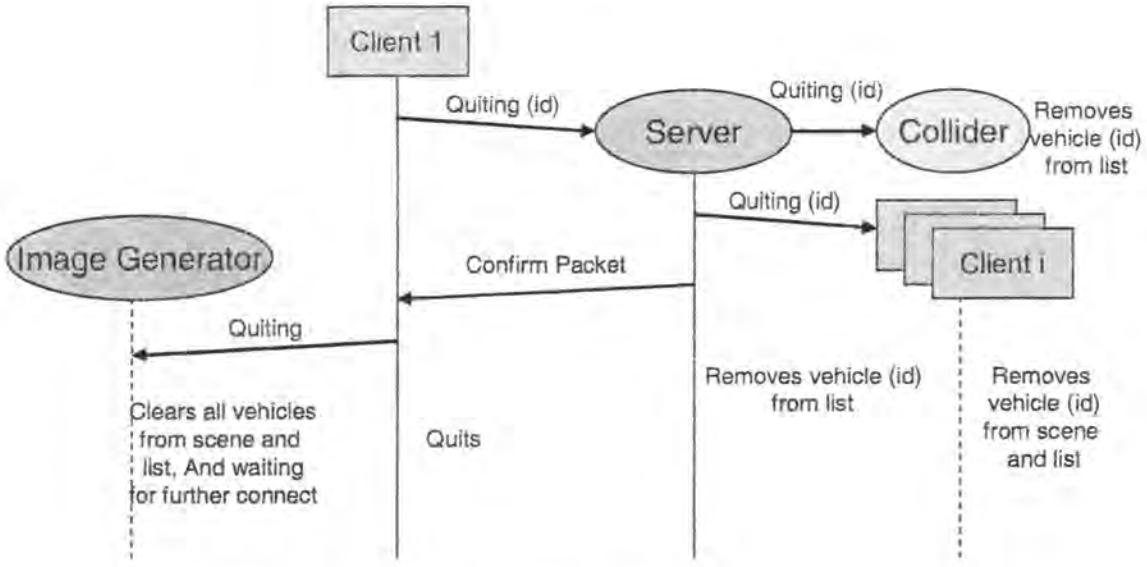

Figure 3-9 Closing Connection

3.3. Architecture Pros and Cons

As stated in the previous sections, the collision algorithm typically resides on the client side is moved to a network server. It is intuitive to think that the collider should be on the client side because the client actually represents a vehicle with which the collider does collision detection and calculation. This is reasonable if only one vehicle is involved. However, if the simulation involves multiple vehicles at different places, then problems arise with this straightforward approach.

For example, consider two vehicles running in the same scene that hit to each other. Suppose both have colliders running and both successfully detect the collision, then both do the collision calculation. Because of the network delay, there is no guarantee that both detect the collision at the same time. The information from the client on the other end of the network is always behind real time. That means both vehicles get different collision responses calculated based on different collision conditions. The level of difference depends on how big the network delay is between these two clients. It is quite possible that client 1 gets the collision result as vehicle 2 hits 1, but client 2 gets opposite result. Figure 3-10 depicts this

case schematically. Both clients catch the collision. But client 1 has a different collision situation than client 2. In the client 1 side of Figure 3-10, vehicle 2 hits vehicle 1 at time t2. But in the client 2 side of this figure at the same time, vehicle 1 hits rear part of vehicle 2. If we trace back to the middle area in Figure 3-10, it shows that vehicle 1 actually hits the front part of vehicle 2. This is correct behavior because both packets used for doing the collision calculation are from the same time point that is t1.

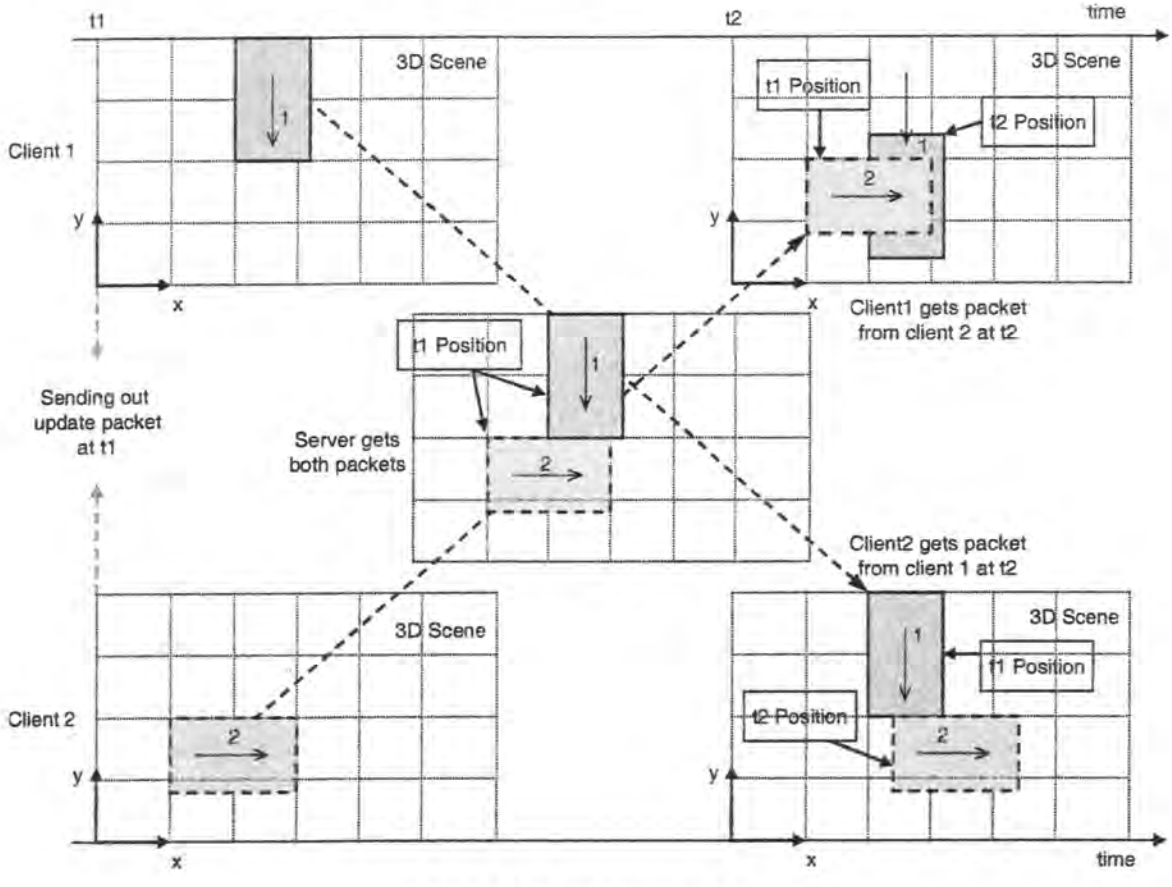

Figure 3-10 Problematic Case

The advantage of moving the collider to a shared network resource is obvious. The information used for collision calculation from different clients is more likely from the same time point, so it is more accurate. Additionally, there is only one collider for all the clients. All collided vehicles get consistent results from the same collision calculation. So the collision

results calculated by one collider for all clients are more reliable, reasonable and consistent than the ones calculated by many colliders from each client. Moreover, collision detection and calculation takes significant time if the number of vehicle is large and the scene model is complicated. We also know that the integration of the VDM the IG take a lot of CPU time when the VDM is accurate (large number of DOF) and the scene model is complicated. It is possible that the simulation can not be run in real time if the collider resides on the clients. So moving the collider to a network server is a way to reduce CPU load on client side, especially when the IG works with VDM on each client.

Nothing is perfect. The collider server introduces network delay that decreases the maximum velocity of vehicles in order to catch and characterize collisions successfully. The details about the performance and its relationship with network delay are explored further in chapter 6.

4. COLLISION DETECTION

It is easy to understand what collision is in the real world. It is a very common physical phenomena that happens anytime, anywhere in our daily life. For a simple example, consider a pencil dropping onto a desk; the pencil collides with desk, bounces, and then comes to rest on top of the desk. When basket ball hits the back board, the ball collides with board, and then the ball bounces back. For a more complex example, when two vehicles crash to each other, they collide, and cause damage. It happens so naturally that we seldom think about when collision happens, and how we might represent it digitally. We never worry about the pencil penetrating the desk and falling to the floor, or the basketball going through the back board and never bouncing back. That is the real world. For the digital world, or virtual world, things are different.

This chapter shows how collision happens between a vehicle and the scene, and between moving vehicles in a virtual world, and how it can be represented. We presume a digital environment exists that consists of some objects (such as walls, buildings, trees, etc), and some vehicles in an interactive 3D virtual environment. The scene is stationary, but the vehicles are not. The goal is to represent realistic behavior when a vehicle hits the scene or another vehicle. Although the collision detection algorithms used in this research are not complicated, they build on some basic common ideas of various well-developed algorithms to provide a reasonable compromise between real-time performance, and realistic physical behavior.

4.1. Background

Collision detection is fundamental in simulation of physical the world, such as virtual assembly, robotics, 3D games, computer animation, physically based modeling, and so on. Different levels of collision detection algorithms are used in different specific fields. The most commonly used algorithms include the Lin-Canny closest features algorithm [9], V-Clip [27],

I-Collide [10], OBB-tree [11, 28]. The other uncommonly used algorithms is V-Collide [12]. Collision algorithm generally falls into one of three categories, based on their primary origins and applications: robotics, computational geometry, and simulation in large-scaled virtual environment.

In the robotics literature, collision detection is generally developed in the context of path planning. The goal is to plan collision-free paths for a robot in a restricted environment by using sophisticated mathematical tools. These are distinguished from simulation-based applications, in which motion is subject to a dynamic and unpredicted environment and can not be formed as a function of time. Collision detection and distance computation are central to the techniques considered in the robotics literature.

For computational geometry, collision detection is used for detecting object interference within a static environment. The objects are either at a fixed location and orientation or move slightly from frame to frame. These techniques are very different from the other two categories.

Simulation in large-scaled virtual environment, like this research, deals with collision detection in a large-scaled scene with hundreds or thousands of polygons, and many moving objects. Even though the scene is fixed, the moving objects are unpredicted. Their motions are affected by external inputs such as force.

Any collision algorithm needs a specific collision model and query type. The collision model is the representation of objects that are collision candidates. Examples of collision models include polygonal models, constructive solid models and parametric surfaces [29]. The complexity of the model determines the complexity of collision detection.
The query type refers to how the algorithm reports collision results. For example, some algorithms compute the penetration distance of colliding objects and/or their separation distance. Other algorithms determine the intersection points and intersection surface normal. In some applications, the details of the collision area are determined.

4.2. Collision model

Before describing the collision detection algorithms implemented in this research, the collision model for vehicles and the scene are presented.

4.2.1. Collision Model for Vehicle

An actual vehicle is comprised of numerous parts characterized by very complicated geometry. It is impossible for a complex geometric model to be used in real time simulation of collision detection. It is also unnecessary – since the collision response is limited to the two-dimensional yaw plane of the vehicle, a simplified model is sufficient for collision detection. The collision implemented in this research is a simply a rectangle with the same width and length as the real vehicle. A typical collision model is shown as Figure 4-1.

The four edges represent the front, rear, left and right sides of the vehicle. The model also incorporates pitch, roll and yaw angles as well the vehicle cg height and dynamics properties, such as position, velocity and acceleration. The additional properties enable the modeling of the vehicle dynamic behavior, and not always in a plane parallel to the world coordinate planes.

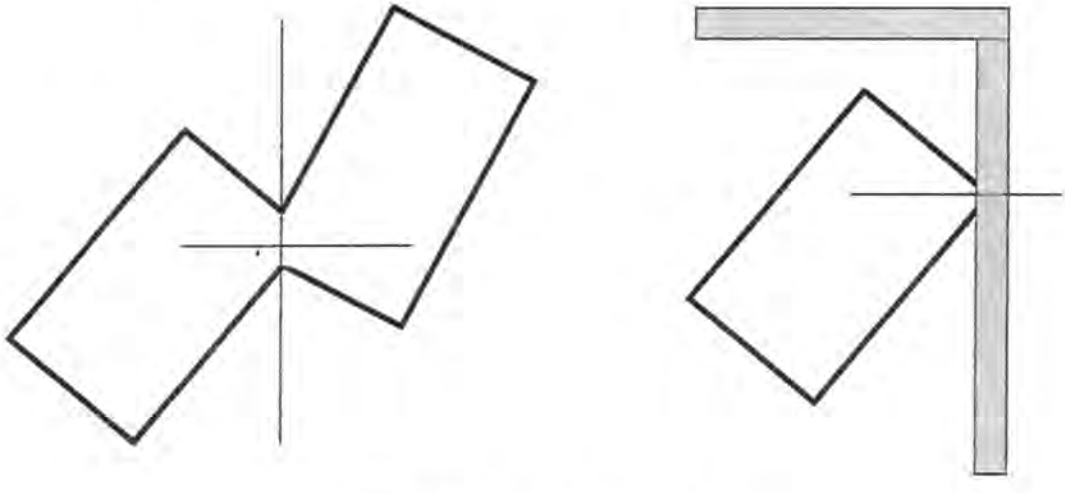

Figure 4-1 Collision Model for Vehicle

4.2.2. Collision Model for Scene

For vehicle-to-scene collision, the Open Scene Graph (OSG) line segment to surface intersection functionality is implemented. Inside the OSG scene graph tree, objects consist of hundreds and thousands of surfaces (polygons). OSG is very efficient at computing the intersection of a line segment with object models. The collision model for the scene is thus a polygonal model.

4.3. Vehicle-to-scene collision detection

As stated in the previous section, the collision model for a vehicle is a rectangle consisting of four line segments. For vehicle-to-scene collision detection, the vehicle model is decomposed into four lines segments. By using OSG internal line segment to surface intersection functionality, vehicle-to-scene collision detection can be done easily with four successive intersection tests.

4.3.1. Line Segment to Surface Intersection Detection

Consider line segment intersection with a convex polygon (most graphics primitives are convex, or can be easily decomposed into sets of convex polygons). The line to surface intersection detection is very efficient by exploiting a hierarchical filter. There are three steps in the intersection process, as shown in Figure 4-2. The first step is to test if the line segment crosses the plane that contains the polygon. If the first step passes, the second step is to compute the intersection point of the line segment and the plane. The last step is to test if the intersection point is inside the polygon. Each step here can be done with several dot products of vectors.

Because the length of the line segment is limited, it fails to pass the first step test for most of polygons. There is no necessary to do the next two steps for most cases. Additionally, by using the hierarchical bounding volume of nodes in scene graph tree, it

dramatically lowers the number of polygons to be tested. So the line segment to convex polygon intersection testing can be done very efficiently by OSG.

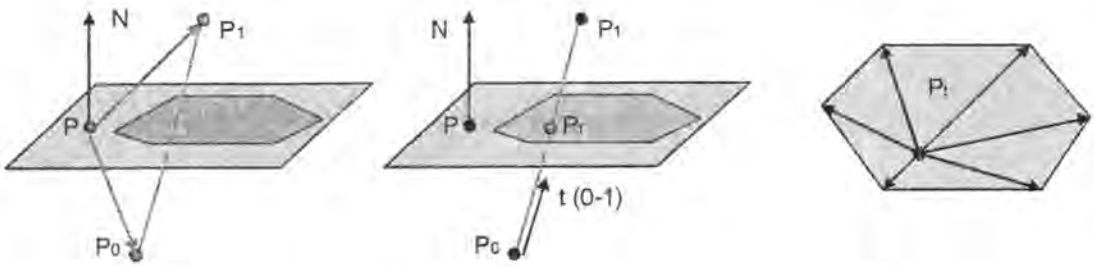

Figure 4-2 Line Segment to Surface Intersection Test

4.3.2. Collision Query

As required by vehicle driving simulation, the collision responses, such as forces, moments, etc, are calculated after calculation detection. To support the collision response, the intersection details must be returned by the collision detection. The normal of the collision surface is needed for determining the directions of the collision and friction forces. The collision points are also necessary for computing collision forces and moments and for deciding which point on the vehicle the collision forces should apply. The penetration distance between two collided objects is not used here.

Once OSG detects any intersection of a line segment to polygons of scene, the intersection points and normal vectors can be obtained easily. There may be several intersection points for a line segment and scene object, as shown Figure 4-3, but only the first point is of interest.

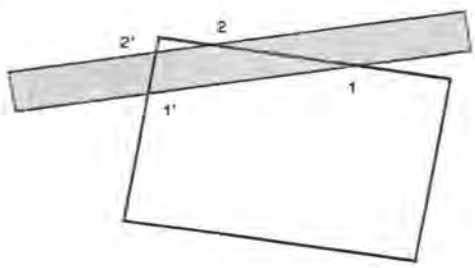

Figure 4-3 Several Intersection Points Case

Usually, two or more vehicle bounding edges intersect with a scene object for each collision. Counting only the first intersection point on each edge, if there are two or more points available after all four bounding line segments have been intersected, then a single representative intersection point must be computed from them. The collision surface normal vector is computed in a similar manner. Figure 4-4 shows the possible cases for vehicle-to-scene collisions.

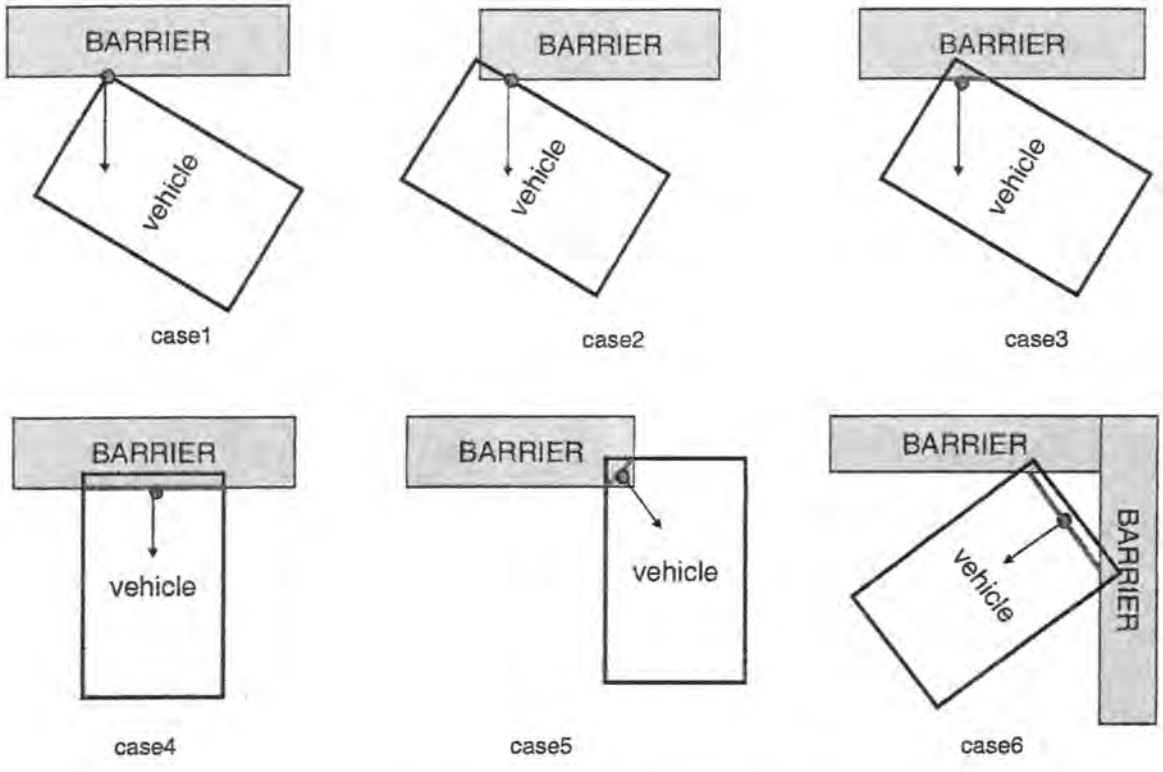

Figure 4-4 Vehicle-to-scene Collision Cases

For one intersection point cases, such as case 1 and 2, the intersection point is taken as the collision point. The normal vector returned from OSG is used as the collision surface normal.

For two intersection point cases, such as case 3, 4 and 5, the mid point of the two computed intersection points is taken as the collision point. The collision normal is the vector that is perpendicular to the line connecting two intersection points and opposite the approaching velocity vector of vehicle to barrier.

For the four intersection-point case, as shown in case 6, there are two steps. First, points are computed at the mid point of the lines defined by the intersection points that are closest to each other. Then the line segment defined by these two points is bisected to get the collision point and collision surface normal.

4.4. Vehicle-to-vehicle collision detection

The algorithm for vehicle-to-vehicle collision detection is simpler than the vehicle-to-scene method because the vehicles are represented by four line segments each. One vehicle collides with another vehicle if any edge of one vehicle intersects with any edge of another one. In order to get collision points and normal and correctly, all the possible intersection tests must be conducted for all the edges. In order to reduce unnecessary calculations a two-step collision detection technique is employed. First, to quickly eliminate cases in which the two vehicles are far apart, a bounding circle proximity filter is implemented. In the second step, the Cohen-Sutherland Clipping Algorithm [21] is used to simplify the intersection test and calculate intersection points.

4.4.1. Bounding Circle

The bounding circle is very efficient for simple collision detection as shown Figure 4-5. To simplify the computation, the cg position of the vehicle is used directly as the center of the bounding circle. The radius is the maximum distance from a rectangle corner to the cg. The condition for continuing to the second step is that the center distance of two bounding circles is less than the sum of their radii.

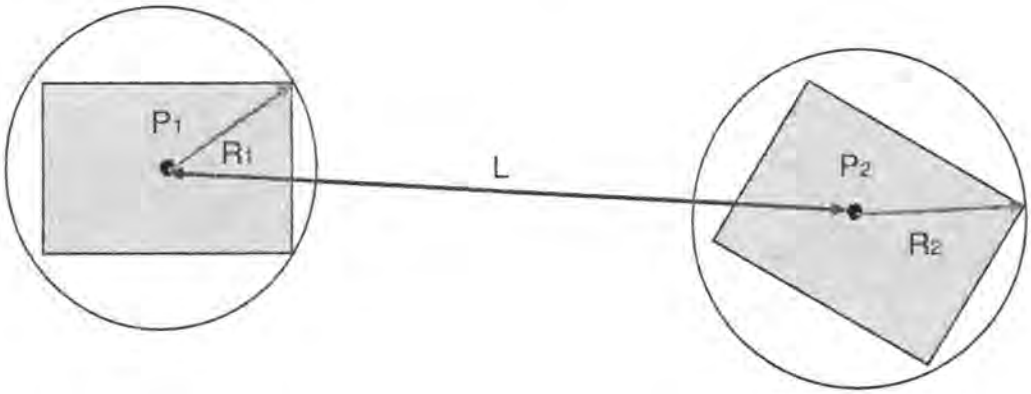

Figure 4-5 Bounding Circle

4.4.2. Cohen-Sutherland Clipping Algorithm

Cohen-Sutherland is a very efficient method for computing the intersection of a line segment with a rectangle. It treats a line as two points, and transforms these two points into the local coordination system of rectangle. As shown in Figure 4-6, the four edges of the rectangle divide the space into nine distinct areas. Each area is represented by a four bit clipping code. The first bit indicates whether a point is located above the top edge. The second bit indicates whether a point is below the bottom edge. The third bit indicates whether point is on the right side of the right edge. The last bit indicates whether point is on the left side of the left edge. If all bits are zero, then the point is inside the rectangle or on the edges. If two points clipping codes have the same bit value equal to 1, then that line is totally outside of the rectangle, such as line l_2 in Figure 4-6. In this case, it is simply rejected. If one point's clipping code is zero, but the other is not, then that line intersects the rectangle, like line l_3 in Figure 4-6. In this case, it is accepted, and the intersection point is computed. For other cases the clipping is done with the left, right, top, and bottom in order to get the new points.

For rectangle to rectangle (or vehicle-to-vehicle) intersection detection, the idea is the same. The coordinate transformation is computed for one vehicle from the world coordination system to the local system of the other vehicle, and the clipping codes for four

corner points of transformed vehicle are computed. Then the Cohen-Sutherland clipping algorithm is applied four times. If a line segment is clipped, then intersection is detected, and an intersection point list is retuned. The intersection points are used for the collision query described in the next section.

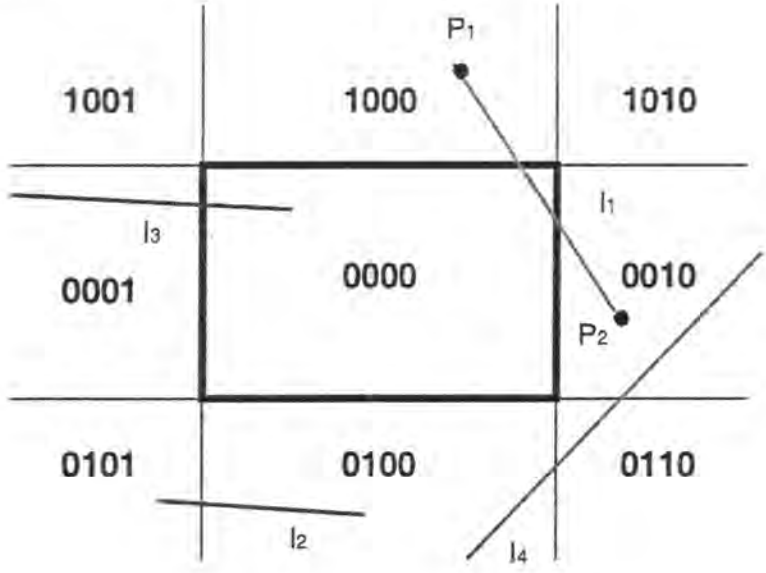

Figure 4-6 Cohen-Sutherland Clipping

4.4.3. Collision Query

Collision response calculations for the vehicle-to-vehicle collision case also require the detection method to return collision points and collision surface normal. Vehicles collide with each other from any possible angle. For various different cases, different approaches are taken to yield reasonable collision points and collision surface normal.

Figure 4-7 shows some general cases for vehicle-to-vehicle collision. All of them have two intersection points. It is simple to compute the collision point that is the mid point of two intersection points, and the impact surface normal that is perpendicular to the impact surface.

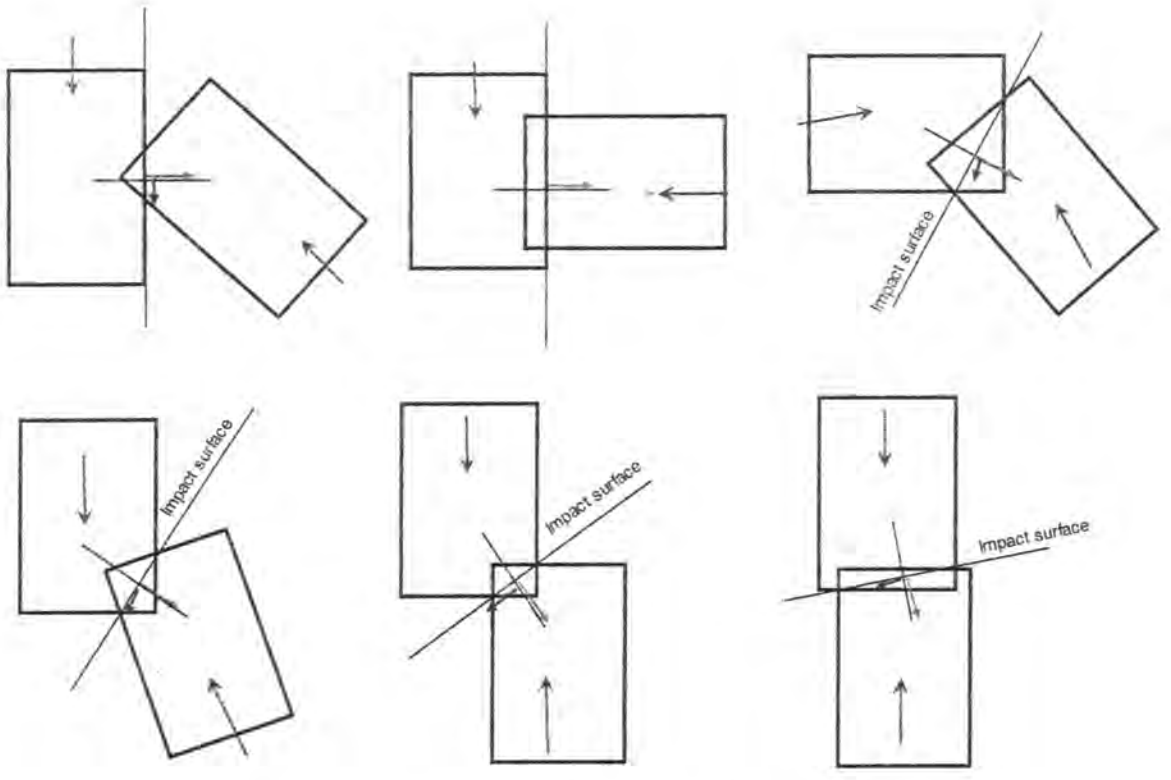

Figure 4-7 Vehicle-to-vehicle Collision – General Cases

Figure 4-8 shows two impossible cases for vehicle-to-vehicle collision. The following example illustrates this situation. Suppose the frame rate is 200Hz (0.005 sec/frame), and the width of vehicle is 5 feet. In order to achieve either case shown in Figure 4-8, the velocity for both vehicles must exceed 5/0.005/2 that is 500 feet/sec (340.9 miles/h or 548.6 km/h). So condition is generally impossible for any typical vehicle. This discussion is focused on simulation within a local network. For long distance network simulation, both cases in Figure 4-8 will be treated as a failed collision detection. This condition will be discussed in chapter 7.

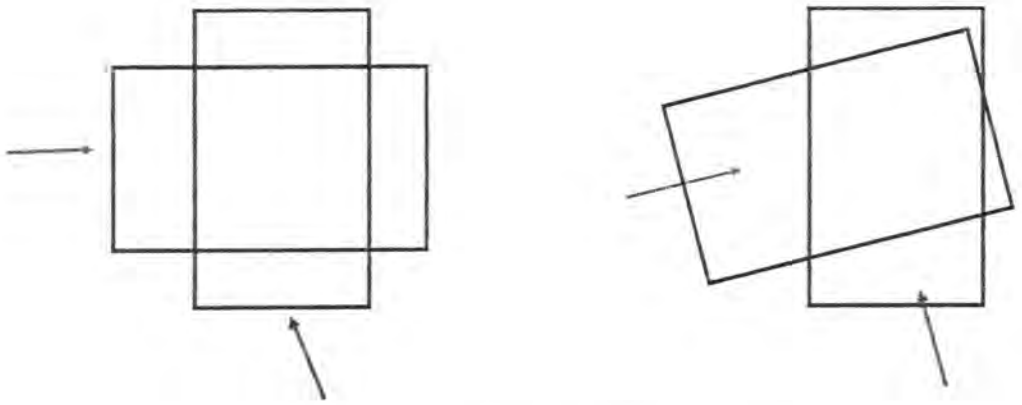

Figure 4-8 Vehicle-to-vehicle Collision – Impossible Cases

Figure 4-9 shows two other impossible cases that have only one intersection point. This condition is not physically impossible, but almost never happens. The approach for dealing with this one intersection point case is to wait until two intersection points appear.

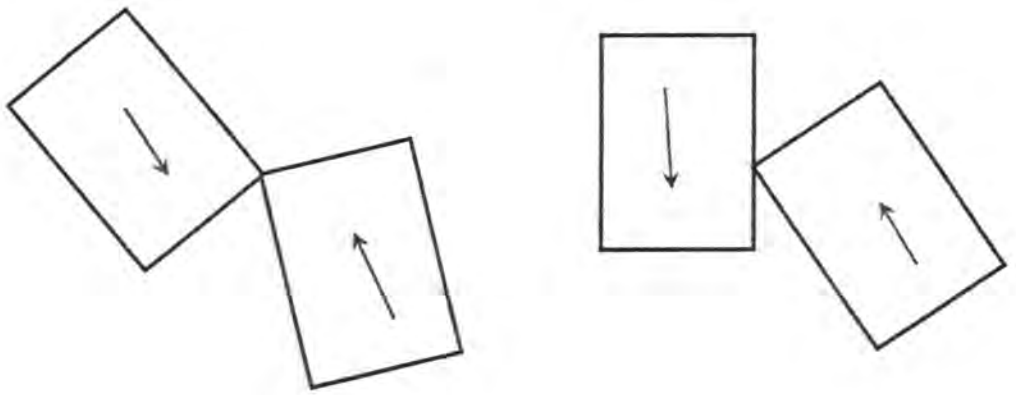

Figure 4-9 Vehicle-to-vehicle Collision –Other Impossible Cases

For the cases shown in Figure 4-10, after the Cohen-Sutherland clipping, four intersection points are obtained. In order to get a single collision point and correct collision surface normal, the four points are divided into two groups according to their relative position. Then all the points in the same group that are supposed to be close to each other are

merged to one point. Finally a single collision point and collision normal can be calculated by using the two intersection point method described in previous section.

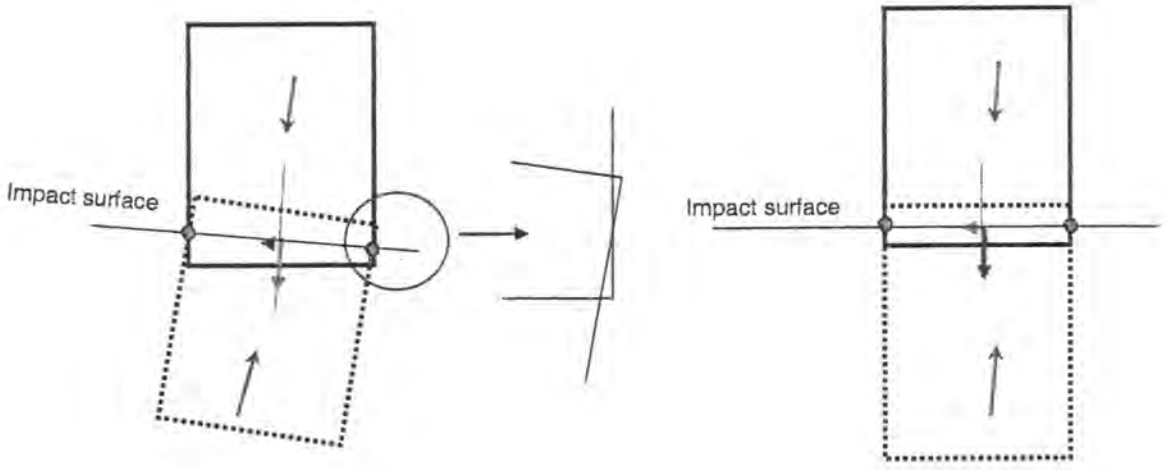

Figure 4-10 Vehicle-to-vehicle Collision – Unusual Cases

5. COLLISION RESPONSE CALCULATION

Based on information obtained from collision detection, the vehicles' parameters and its dynamic state, this chapter presents algorithms for determining realistic dynamic collision response in real time. The vehicle-to-vehicle collision response algorithm is introduced in the first section. It has two parts: momentum conservation and coefficient of restitution. Although the energy loss method is not used in this thesis, it is introduced as a reference for the coefficient of restitution technique. Several important coefficients used in the collision response algorithms will be pointed out in order to show how their values are determined in different collision situations. At last, a vehicle-to-scene collision response algorithm is described briefly as a specialization of the general vehicle-to-vehicle collision response algorithms.

5.1. Vehicle-to-vehicle Collision Algorithms

The goal of the collision response algorithm is to make the collision simulation reasonable and correct for a collaborative virtual environment, but not necessarily accurate in engineering detail. As explained in chapter 4, the vehicle model is simplified as a rectangle consisting of four line segments. Further assumptions are made in this research to simplify vehicle collision simulation, namely:

- There is only one collision at a time for one vehicle.
- Collision force is a constant value throughout the same collision duration that lasts only one frame time.
- Collision forces and moments are applied in the vehicle yaw plane.
- The collision surface normal vector is in the vehicle yaw plane.
- For vehicle-to-vehicle collision, the yaw planes of two vehicles are the same.

Also, many symbols, conventions, and various coordinate systems are used to help explain and implement the collision simulation algorithms in this chapter. Table 5-1 shows all

the symbols used in deriving of the collision response equations. Some of them will be referred to as vehicle specific by adding subscripts and superscript.

Table 5-1 Vehicle Dynamics Symbols

No.	Symbol Name	Unit	Full Name
1	V	feet/sec	Velocity of vehicle at cg (vector)
2	R	rad/sec	yaw rate
3	p (lower case)	feet/sec	Approaching velocity at collision point of two impact bodies
4	e	N/A	Coefficient of Restitution
5	μ	N/A	Coefficient of collision friction
6	P (upper case)	N/A	Factor of energy loss
7	M	lbs	Mass of vehicle
8	Izz	$lbs \cdot feet^2$	Moment of inertial
9	I	$lbs \cdot \sec$	Impulse of collision
10	W	feet	Width of vehicle
11	l	feet	Wheel base
12	w	feet	Track
13	h	feet	Cg height
14	P	feet	Vector from collision point to vehicle cg

And the conventions are:

- All the variables used take sign.
- The subscript '1' means vehicle 1, '2' means vehicle 2, e.g., m_1 means mass of vehicle1, and m_2 means mass of vehicle2

- The subscript 'x' means x component, 'y' means y component, in the corresponding coordinate system, e.g., V_{1x} means the "x" component of velocity of vehicle 1. Variables do not contain symbols to indicate their coordinate system.
- The superscript 'p' means post collision, 'a' means pre (ante) collision, e.g., V_{2y}^{a} means y component of velocity of vehicle 2 at the moment before collision.

There are three kinds of coordinate systems including the World Coordinate System (WCS), Vehicle Local Coordinate System (VCS), and Collision Coordinate System (CCS). Figure 5-1 shows the differences.

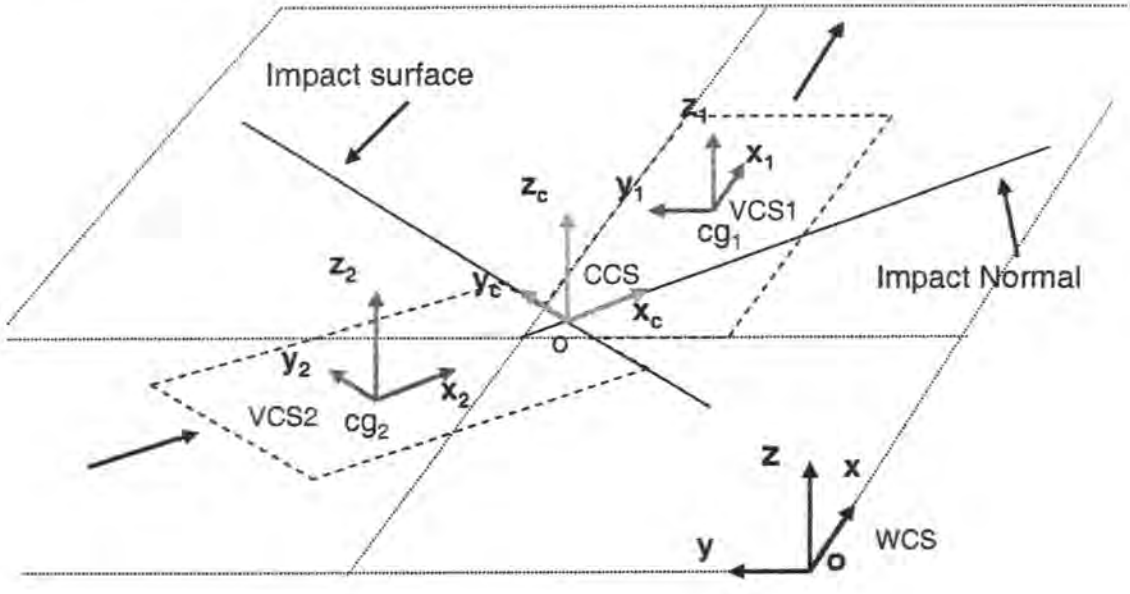

Figure 5-1 Coordination Systems

- All coordinate systems obey the right-hand rule, and z is always upward.
- In VCS, x is forward. So positive y is toward the left.
- For CCS, the intersection line of the impact surface and vehicle yaw plane is the y-axis. The line that goes through the impact point and is perpendicular to the impact surface is the x-axis. Given two colliding vehicles 1 and 2, positive x always points to the side on which vehicle 1 stays. In other words, the x-value of vehicle 1's cg is

always positive in CCS. All variables will be transformed into CCS for the collision response calculation.

In order to remove the time variable from the collision response algorithms, the impulse is used instead of force. Basically, impulse is the integration of force upon time. That is:

$$I = \int_{t_0}^{t_1} F dt$$

Since F is assumed constant through out collision, then

$I = F * \Delta t$,

Where Δt is collision duration.

The advantage of using the impulse is that it does not require the collision duration in the collision response calculation to have the same value as the integration time step during vehicle dynamics integration.

5.1.1. Basic Dynamics Properties

If two objects are not totally elastic, then any collision between them causes energy loss. But the total momentum is always maintained throughout the collision. To calculate how much the energy is lost, two approaches are developed here: coefficient of restitution and kinetic energy loss. For either one, the basic dynamics equations are the same.

As shown in Figure 5-2, suppose collision occurs between two vehicles (vehicle 1 and 2). And the collision point and collision normal vector are all known. $\vec{\rho}_1$ $(a_1 + b_1 i)$ and $\vec{\rho}_2$ $(a_2 + b_2 i)$ represent vectors from the cg's of the corresponding vehicles to the collision point. Many other variables are known before collision. They are: m_1, Izz_1, m_2, Izz_2, V_{1x}^a, V_{1y}^a, V_{2x}^a, V_{2y}^a, r_1^a, and r_2^a. The unknown variables are: I_{1x}, I_{1y}, I_{2x}, I_{2y}, V_{1x}^p, V_{1y}^p, V_{2x}^p, V_{2y}^p, r_1^p, and r_2^p.

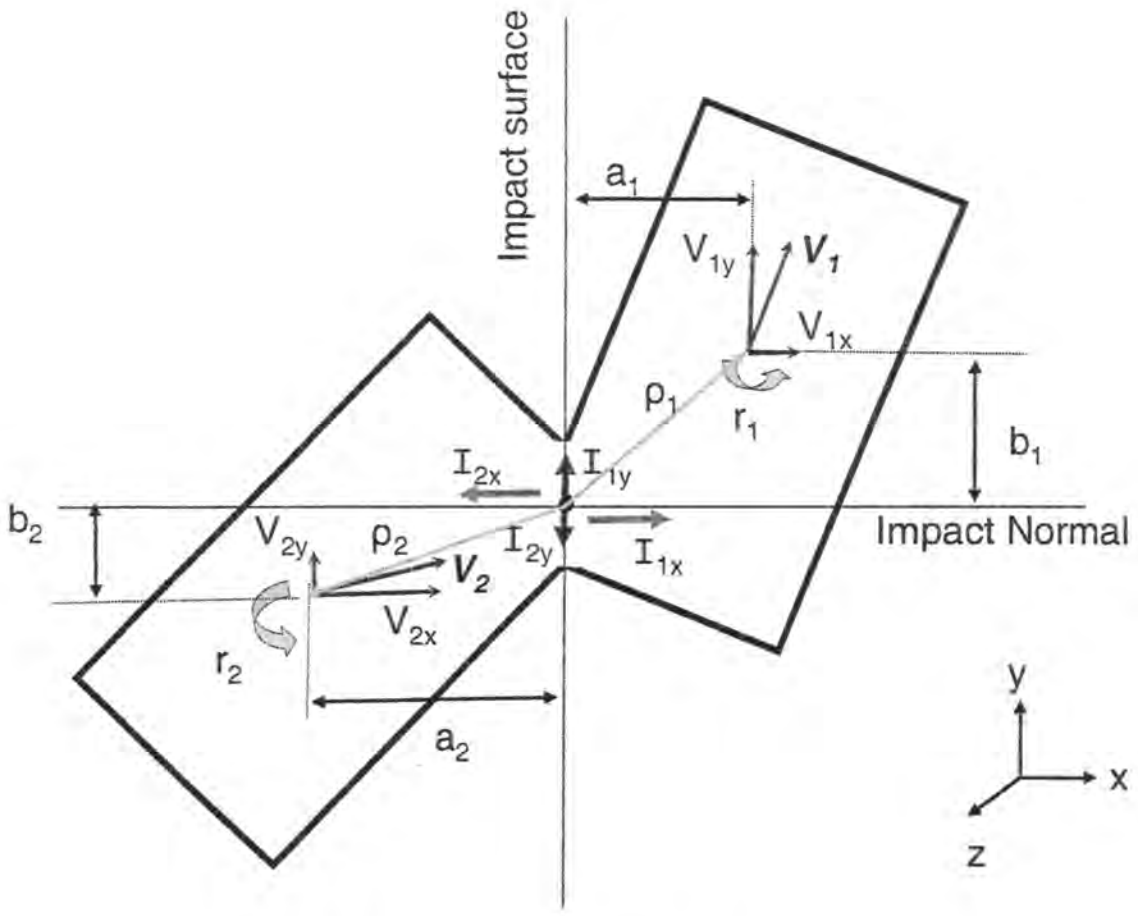

Figure 5-2 Vehicle-to-vehicle Collision

The linear momentum equations for vehicle 1 and vehicle 2 are:

$$m_1(\vec{V}_1^{\,p} - \vec{V}_1^{\,a}) = \vec{I}_1 \qquad (5\text{-}1)$$

$$m_2(\vec{V}_2^{\,p} - \vec{V}_2^{\,a}) = \vec{I}_2 \qquad (5\text{-}2)$$

The angular momentum equations for vehicle 1 and vehicle 2 are:

$$Izz_1(r_1^p - r_1^a) = \vec{I}_1 \times \vec{\rho}_1 \qquad (5\text{-}3)$$

$$Izz_2(r_2^p - r_2^a) = \vec{I}_2 \times \vec{\rho}_2 \qquad (5\text{-}4)$$

Replacing the vector in equations 5-1, 5-2, 5-3 and 5-4 with their x and y components in the CCS, yields:

For vehicle 1:

$$m_1(V_{1x}^p - V_{1x}^a) = I_{1x} \quad (5\text{-}5)$$

$$m_1(V_{1y}^p - V_{1y}^a) = I_{1y} \quad (5\text{-}6)$$

$$Izz_1(r_1^p - r_1^a) = I_{1x}b_1 - I_{1y}a_1 \quad (5\text{-}7)$$

For vehicle 2:

$$m_2(V_{2x}^p - V_{2x}^a) = I_{2x} \quad (5\text{-}8)$$

$$m_2(V_{2y}^p - V_{2y}^a) = I_{2y} \quad (5\text{-}9)$$

$$Izz_2(r_2^p - r_2^a) = I_{2x}b_2 - I_{2y}a_2 \quad (5\text{-}10)$$

During the same collision, the absolute values of collision forces or impulses for two vehicles should be the same, but with opposite sign. The relationships are:

$$\vec{I}_2 = -\vec{I}_1 \quad (5\text{-}11)$$

Or

$$I_{2x} = -I_{1x} \quad (5\text{-}12)$$

$$I_{2y} = -I_{1y} \quad (5\text{-}13)$$

The x- and y-components for each impulse are related by:

$$I_{1y} = \mu I_{1x} \quad (5\text{-}14)$$

Where μ is coefficient of vehicle collision friction that varies with different collision situations. (Details on the friction model are covered in the next section of this chapter.) Its value ranges from 0 to a positive value (μ_{max}=0.3) that is less than 1. Substituting I_{1y} into equation 5-13 yields:

$$I_{2y} = -\mu I_{1x} \quad (5\text{-}15)$$

Substituting equations 5-12, 5-14 and 5-15 into 5-5, 5-6, 5-7, 5-8, 5-9 and 5-10, yields:

$$m_1(V_{1x}^p - V_{1x}^a) = I_{1x} \quad (5\text{-}16)$$

$$m_1(V_{1y}^p - V_{1y}^a) = \mu I_{1x} \quad (5\text{-}17)$$

$$Izz_1(r_1^p - r_1^a) = I_{1x}b_1 - \mu I_{1x}a_1 \quad (5\text{-}18)$$

$$m_2(V_{2x}^p - V_{2x}^a) = -I_{1x} \quad (5\text{-}19)$$

$$m_2(V_{2y}^p - V_{2y}^a) = -\mu I_{1x} \quad (5\text{-}20)$$

$$Izz_2(r_2^p - r_2^a) = -I_{1x}b_2 + \mu I_{1x}a_2 \quad (5\text{-}21)$$

Representing the unknown values with given input values and the impulse for equations 5-16 to 5-21, results in:

$$V_{1x}^p = V_{1x}^a + I_{1x}/m_1 \quad (5\text{-}22)$$

$$V_{1y}^p = V_{1y}^a + \mu I_{1x}/m_1 \quad (5\text{-}23)$$

$$r_1^p = r_1^a + (I_{1x}b_1 - \mu I_{1x}a_1)/Izz_1 \quad (5\text{-}24)$$

$$V_{2x}^p = V_{2x}^a - I_{1x}/m_2 \quad (5\text{-}25)$$

$$V_{2y}^p = V_{2y}^a - \mu I_{1x}/m_2 \quad (5\text{-}26)$$

$$r_2^p = r_2^a + (-I_{1x}b_2 + \mu I_{1x}a_2)/Izz_2 \quad (5\text{-}27)$$

So far, there are six equations, but seven unknown variables. Another equation, developed in either one of the next two sections, is needed to obtain a solution.

5.1.2. Kinetic Energy Loss

Any collision causes energy loss. So the kinetic energy loss method introduced in this section is a candidate for this purpose. Suppose the kinetic energy before collision is E^a, and the energy after collision is E^p. Then there exists the following relationship between them:

$$E^p = PE^a \qquad (5\text{-}28)$$

Where P is the energy loss factor that ranges from 0 to a positive value that is less than 1, and

$$E^a = (m_1(V_{1x}^a)^2 + m_1(V_{1y}^a)^2 + m_2(V_{2x}^a)^2 + m_2(V_{2y}^a)^2 + Izz_1(r_1^a)^2 + Izz_2(r_2^a)^2)/2 \qquad (5\text{-}29)$$

$$E^p = (m_1(V_{1x}^p)^2 + m_1(V_{1y}^p)^2 + m_2(V_{2x}^p)^2 + m_2(V_{2y}^p)^2 + Izz_1(r_1^p)^2 + Izz_2(r_2^p)^2)/2 \qquad (5\text{-}30)$$

Substitute all the post collision variables from equation 5-22 to 5-27, 5-29, and 5-30 into equation 5-28 and simplify, to get:

$$\begin{aligned}
&(I_{1x})^2\left(\frac{1+\mu}{m_1} + \frac{(b_1 - \mu a_1)^2}{Izz_1} + \frac{1+\mu}{m_2} + \frac{(b_2 - \mu a_2)^2}{Izz_2}\right) \\
&- 2I_{1x}(-V_{1x}^a - \mu V_{1y}^a + r_1^a(\mu a_1 - b_1) + V_{2x}^a + \mu V_{2y}^a + r_2^a(b_2 - \mu a_2)) \\
&+ (1-P)(m_1((V_{1x}^a)^2 + (V_{1y}^a)^2) + Izz_1(r_1^a)^2 + m_2((V_{2x}^a)^2 + (V_{2y}^a)^2) + Izz_2(r_2^a)^2) \\
&= 0
\end{aligned} \qquad (5\text{-}31)$$

To get the collision impulse, we have to solve above quadric equation. Thus it is possible that there are two roots, or no root at all. As seen from equation 5-31, there are some potential problems. They include:

- The computation is not simple.
- Under some collision conditions, there is no root.
- If two roots if they exist, one must be chosen.

Due to these challenges, the kinetic energy loss method is not adopted in this research. Instead, the coefficient of restitution method is used, which is introduced in the next section.

5.1.3. Coefficient of Restitution

The coefficient of restitution method is a collision simulation algorithm based on the theory that the separating velocity of two collided objects at the collision point is always less than the approaching velocity at the collision point. Here, the approaching velocity and separating velocity of two objects means the normal component of relative velocity of two objects. Figure 5-3 shows a simple example of relative velocity, approaching velocity, collision surface and collision normal.

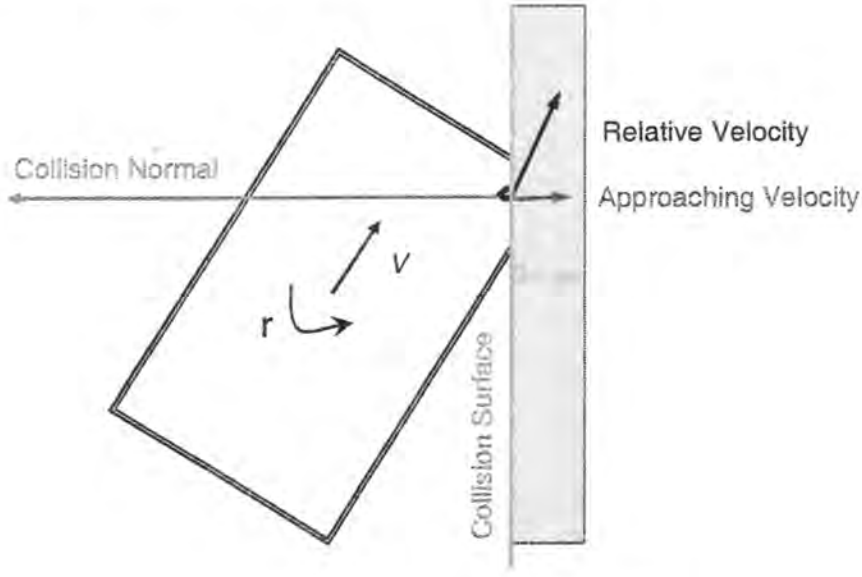

Figure 5-3 Approaching Velocity

If p represents the approaching velocity, then the relationship between pre-collision and post-collision states is:

$$p^p = -ep^a \qquad (5\text{-}32)$$

Where e is coefficient of restitution (generally positive and less than 1, but in this research, it can be negative in some special cases). These special conditions for e are described in the next section of this chapter.

As shown in Figure 5-2, using the vehicle velocity and yaw rate to represent p yields:

$$p^a = V_{2x}^a - V_{1x}^a + r_2^a b_2 - r_1^a b_1 \qquad (5\text{-}33)$$

$$p^p = V_{2x}^p - V_{1x}^p + r_2^p b_2 - r_1^p b_1 \qquad (5\text{-}34)$$

Substituting equations 5-33 and 5-34 into equation 5-32, gives:

$$V_{2x}^p - V_{1x}^p + r_2^p b_2 - r_1^p b_1 = -e(V_{2x}^a - V_{1x}^a + r_2^a b_2 - r_1^a b_1) \qquad (5\text{-}35)$$

Replacing all post-state variables in equation 5-35 with the ones in equations 5-22 through 5-27, results in:

$$\begin{aligned} &-e(V_{2x}^a - V_{1x}^a + r_2^a b_2 - r_1^a b_1) \\ &= (V_{2x}^a - I_{1x}/m_1) - (V_{1x}^a + I_{1x}/m_2) \\ &+ (r_2^a + (-I_{1x}b_2 + \mu I_{1x}a_2)/Izz_2)b_2 - (r_1^a + (I_{1x}b_1 - \mu I_{1x}a_1)/Izz_1)b_1 \end{aligned} \qquad (5\text{-}36)$$

The simplified version of the above equation is:

$$I_{1x}\left(\frac{1}{m_1} + \frac{(b_1 - \mu a_1)b_1}{Izz_1} + \frac{1}{m_2} + \frac{(b_2 - \mu a_2)b_2}{Izz_2}\right) = (1+e)(V_{2x}^a + r_2^a b_2 - (V_{1x}^a + r_1^a b_1)) \qquad (5\text{-}37)$$

Let,

$M_1 = \frac{1}{m_1} + \frac{(b_1 - \mu a_1)b_1}{Izz_1}$, and $M_2 = \frac{1}{m_2} + \frac{(b_2 - \mu a_2)b_2}{Izz_2}$ represent the reciprocal of mass for vehicle 1 and vehicle 2.

And let,

$VP_{1x}^a = V_{1x}^a + r_1^a b_1$, and $VP_{2x}^a = V_{2x}^a + r_2^a b_2$ present velocities at the collision point for vehicle 1 and 2 respectively.

Combining and simplifying yields:

$$I_{1x} = \frac{(1+e)(VP_{2x}^a - VP_{1x}^a)}{M_1 + M_2} \qquad (5\text{-}38)$$

In equation 5-38, the term $VP_{2x}^a - VP_{1x}^a$ is actually the approaching velocity of two vehicles. So in the final equation it turns out that impulse is related only to approaching velocity and the vehicles' mass. The equation 5-38 is very simple to implement for computing the collision impulse. Also it is very easy to implement the passing through cases for vehicle-

to-vehicle collision by setting e to a negative value. These details are covered in the next section.

5.2. Collision Coefficient Models

5.2.1. Collision Friction Model

According to the classic Coulomb friction model, if the relative tangent velocity of two contacted objects is not zero, the friction force between them is proportional to the normal force (μf_N). If the relative tangent velocity is zero, the friction force is equal to tangential force that should be less than $\mu_s f_N$, where μ_s is the coefficient of static friction. For vehicle collision simulation, it is impossible to implement static collision friction force by using Coulomb's friction model. Because there is no way to get the external tangential force to compute the static friction force. The collision calculation is done in one single frame time. No acceleration is involved. So the friction force is set to 0 if there is no relative tangent velocity between two objects. However, if there is no static friction force in the collision simulation it can cause problems. Consider the following example depicted in Figure 5-4.

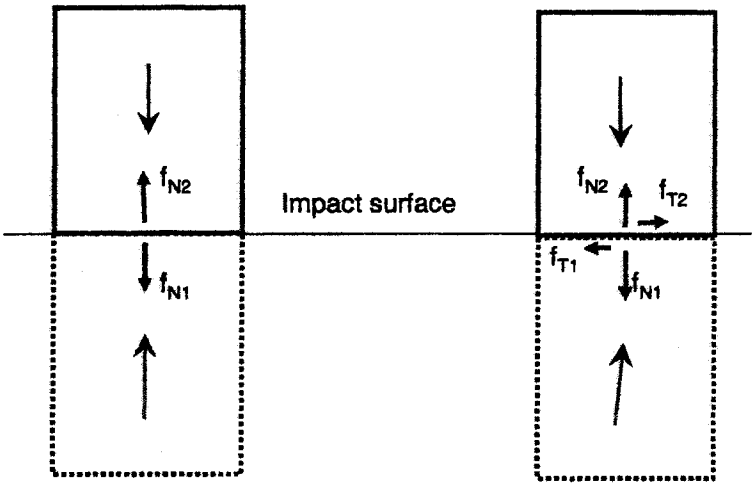

Figure 5-4 Collision Friction

The left collision case is head to head collision. Because there is no relative velocity along the impact surface, the friction force is 0. However, for the collision case on the right-

hand side of Figure 5-4, because the relative tangent velocity is not zero, the friction force is $f_T = \mu f_N$, even though the relative tangent velocity is very, very tiny. During the simulation, when the second collision case happens, the vehicles spin apart from each other very quickly and the resulting behavior appears unrealistic.

To address this challenge, a new collision friction model is introduced as shown in Figure 5-5. The new model introduces a linear ramp so that small values of approaching angle (α) yield relatively small friction force contribution. As shown in Figure 5-5, α is the angle between relative velocity of the two objects and the collision normal as defined in chapter 4. The relative velocity is the difference of the velocity of vehicle 2 from the velocity of vehicle 1. In this research, trial and error testing indicated a reasonable threshold value at which the approach angle ramp stops is 10 degrees. The ramp in the above model simulates the tangential impulse. The tangential impulse increases gradually along with approaching angle.

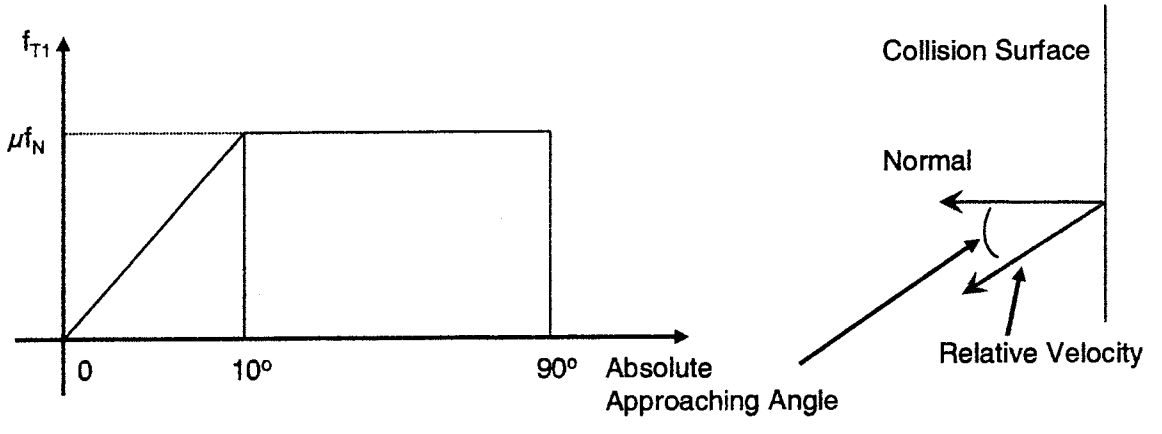

Figure 5-5 Collision Friction Model

5.2.2. Coefficient of Restitution Model

The coefficient of restitution is another coefficient used in vehicle collision response algorithm. This coefficient directly determines how vehicles behave after collision. Vehicles may bounce back from each other, or go through each other. The response velocity may be large or small comparing to the velocity before the collision. All of these behaviors are

determined by coefficient of restitution (e). So, what determines the e value? Basically, the e value is determined by the approach angle (α), relative velocity and collision point. The approach angle and relative velocity have been defined in previous sections.

According to Macmillan [22], typical vehicle collisions have coefficients of restitution of between 0.05 and 0.3. For small approach angle, the lower value is a better approximation. For large approach angle, the larger value is better. Based on this, it is useful to set the coefficient of restitution as a function of the approach angle. It is defined here as:

$$e(\alpha) = 0.175 - 0.125\cos(2\alpha) \qquad (5\text{-}39)$$

Figure 5-6 shows e as a function of α.

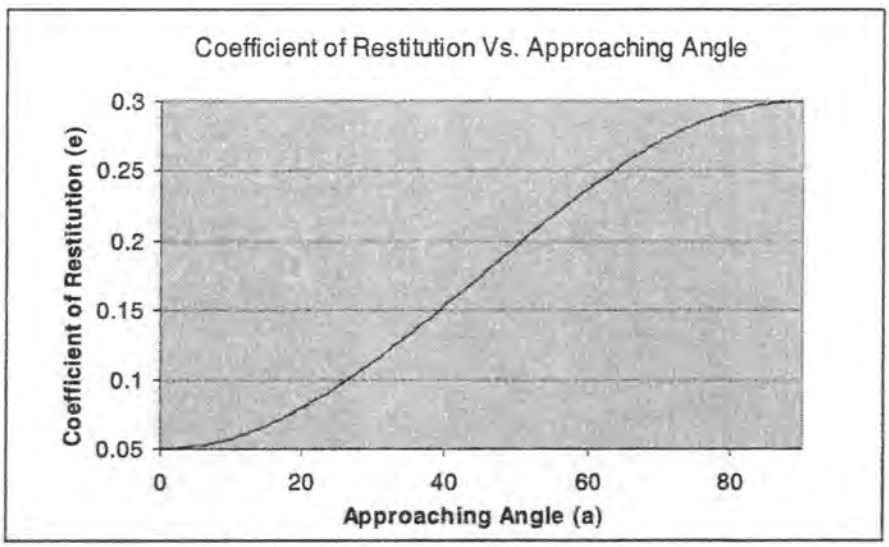

Figure 5-6 Coefficient of Restitution vs. Approaching Angle

The model shown above is for general collision cases in which the vehicles bounce back after collision. It is very useful to implement collision cases in which the vehicles pass through each other, as in a glancing blow, when the collisions are not major.

Figure 5-7 shows how such a minor collision is defined. There is a central box defined inside the vehicle. If the vehicle relative velocity, starting from the collision point, runs into

the vehicle central box, it is considered a major collision. Otherwise, it is a minor collision. For a minor collision, the e value is set to a negative value, then the two collided objects will not separate from each other. Instead, they get closer to each other, but at a smaller approaching speed.

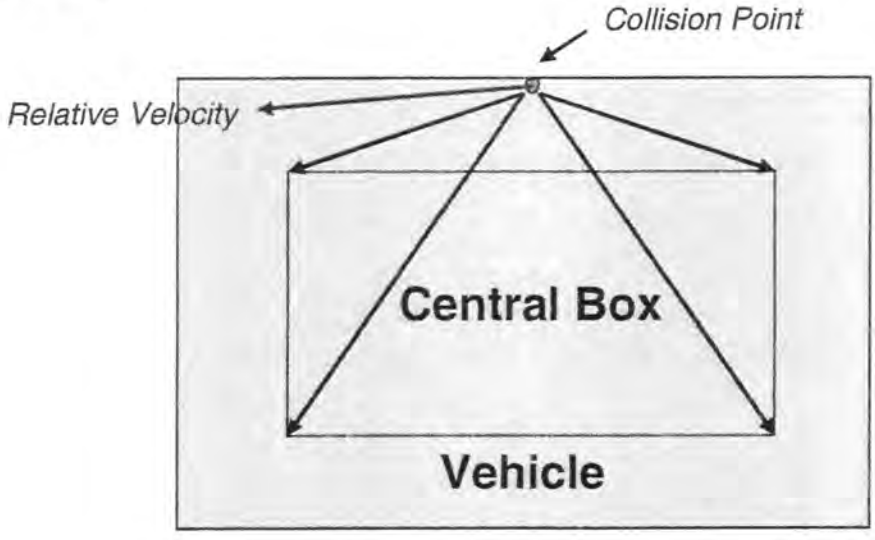

Figure 5-7 Major or Minor Collision

Figure 5-7 also illustrates the basic idea of how to detect the minor collision. If the collision point falls outside the central box, then check if all vectors from the collision point to the four corners of central box sit on one side of relative velocity vector. The cross product of the relative velocity with all four vectors will do the job. If all the cross product values have the same sign, then it is a minor collision. Otherwise, it is a major collision.

5.3. Specializations of Vehicle-to-Vehicle Collision

By using the coefficient of restitution method, it is very convenient to simulate different collision scenarios, such as completely elastic objects or totally inelastic objects. There are, however, two other interesting collision cases: vehicle-to-scene and vehicle head-to-head collisions.

5.3.1. Vehicle-to-scene Collision

Because any scene object is stationary it can be treated as a vehicle whose mass is infinite and velocity is zero. By treating scene object as vehicle 2, a special case of equation 5-38 is (vehicle-to-scene collision response):

$$M_2 = \frac{1}{m_2} + \frac{(b_2 - \mu a_2)b_2}{Izz_2} = 1/\infty + 0 = 0, \text{ and}$$

$$VP_{2x}^a = V_{2x}^a + r_2^a b_2 = 0, \text{ so}$$

$$I_{1x} = \frac{(1+e)(VP_{2x}^a - VP_{1x}^a)}{M_1 + M_2} = \frac{-VP_{1x}^a(1+e)}{M_1} \qquad (5\text{-}40)$$

5.3.2. Head to Head Collision

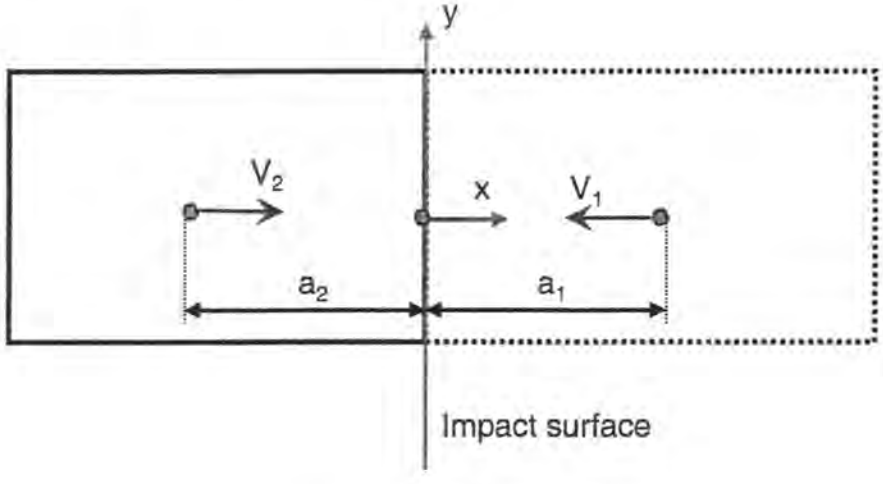

Figure 5-8 Head to Head Collision

Vehicle head-to-head collision is a special case that is perhaps the simplest one. As shown in Figure 5-8, $V_{1x}^a = V_1$, $V_{1y}^a = 0$, $V_{2x}^a = V_2$, $V_{2y}^a = 0$, $r_1^a = 0$, $r_2^a = 0$, and $b_1 = b_2 = 0$. There is no collision friction force because the relative tangential velocity is zero. So μ=0. Substituting these values into equation 5-38, the result in:

$$I_{1x} = \frac{(1+e)(V_2 - V_1)}{1/m_1 + 1/m_2} \qquad (5\text{-}41)$$

5.4. Collision Responses

After computing the normal impulse for vehicle 1 (I_{1x}), all the other impulses can be calculated from equations 5-12, 5-14, and 5-15. Thus it is simple to calculate the forces and moment applied at the cg point for each vehicle. Hereinafter, subscript 'i' is used to represent either vehicle 1 or vehicle 2. Suppose the integration time step for VDM is Δt, then the forces will be:

$$F_{ix} = I_{ix} / \Delta t \qquad (5\text{-}42)$$

$$F_{iy} = I_{iy} / \Delta t = \mu I_{ix} / \Delta t \qquad (5\text{-}43)$$

$$M_i = (F_{ix} b_i - F_{iy} a_i) = (b_i - \mu a_i) I_{ix} / \Delta t \qquad (5\text{-}44)$$

All the other post collision state values, such as V_{1x}^p, V_{1y}^p, V_{2x}^p, V_{2y}^p, r_1^p, and r_2^p, can be calculated using equations 5-22 to 5-27. After completing the collision response calculation for all the vehicles, the collider sends out the collision packet that includes all the collision responses to the corresponding client (VDM) via the network server.

6. VEHICLE DYNAMICS IMPLEMENTATION

The vehicle dynamics model (VDM) is a central component of any vehicle driving simulation. The VDM integrates user inputs based on a model of vehicle parameters to produce the real time response of the simulated vehicle. The choice of dynamics engine is the central factor in balancing the realism and performance of the simulation. As introduced in chapter 3, for this work two different vehicle dynamics models were used: a simple 7 DOF model and a 17 DOF commercial dynamics simulation know as Vehicle Dynamics Analysis Nonlinear (VDANL).

6.1. VDANL Implementation

VDANL is commercial software simulation produced by Systems Tech, Incorporated. VDANL provides an application programmers interface (API) that facilitates integration of VDANL with a user's program. This section focuses on how to set up the VDANL to make it work with IG and Collider Server.

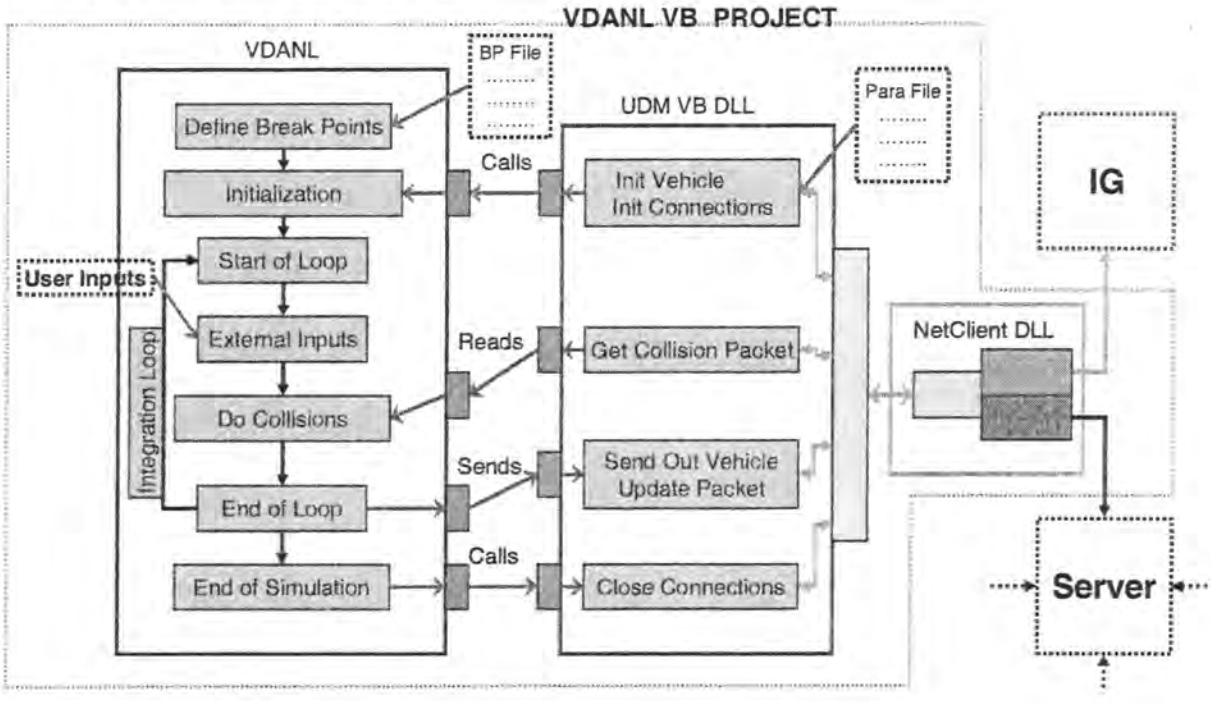

Figure 6-1 VDANL Architecture

When VDANL receives collision responses from the server, not all the response values are used. VDANL does not allow the user to overwrite its internal state values, so the application programmer cannot replace VDANL computed velocities and yaw-rates with ones computed by the collision server. Instead of direct replacement of these values, the application programmer can influence the simulation frame through means of a user defined module (UDM) that is called by VDANL at a specific time during each simulation frame. The capabilities of UDM are limited. They include initializing vehicle parameters and initial states, querying values, and getting external inputs, such as collision forces, moment, and tire positions. Figure 6-1 shows VDANL architecture and relationship among VDANL, UDM, networking code, IG and Collider Server.

The flow of the VDANL simulation process is depicted in the leftmost box. This flow is broken into three stages: initialization, dynamics integrations, and end of simulation.

6.1.1. Initialization

During VDANL initialization, the VDANL process calls the UDM, to allow it to initialize. The UDM reads in the initial vehicle parameters from a parameter file and returns them to VDANL. The UDM also initializes the connections to both the image generator and the collider server. These connections are utilized by the UDM in subsequent calls to it from the Dynamics Integration stage of the VDANL process.These calls allow the UDM to receive packets from the collider server and send out vehicle update information during the integration loop.

6.1.2. Dynamics Integrations

The dynamics integration loop is the core of the VDANL simulation process. At each step of the simulation, prior to state integration any collision forces and moments received from the collider server through a callback to the UDM. Once the integration step is finished,

another call to the UDM allows the newly computed vehicle state to be sent to the collider server and IG.

6.1.3. End of Simulation

When the simulation run terminates, a final call to the UDM closes the connections to the IG and Collider Server.

6.2. Simplified VDM Implementation

The simplified VDM was written in C++ specifically for the present work. It's architecture is far simpler than VDANL and as a result is much easier to integrate with the other system components. The architecture of the simplified system is shown in Figure 6-2.

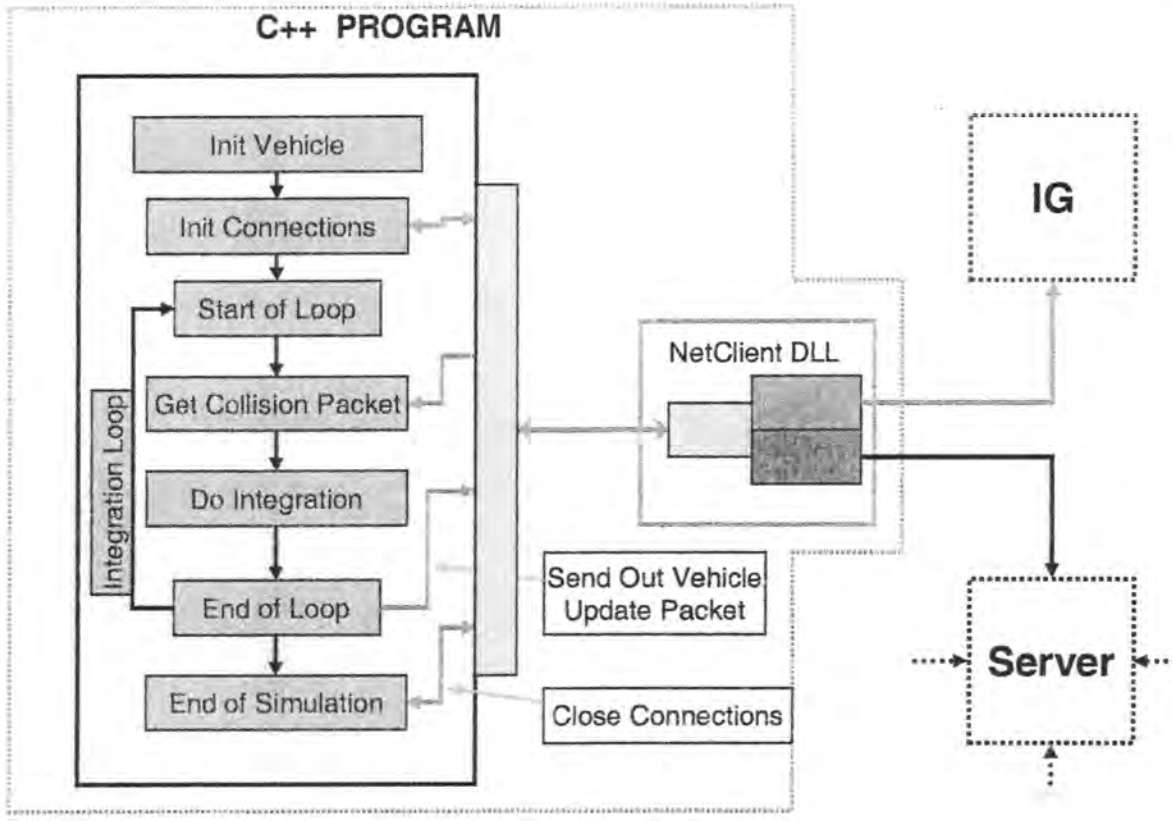

Figure 6-2 Simplified VDM Architecture

The other system components in the simplified system are the same as in the VDANL case. The VDM integration processes are similar too, but the communications between VDM and networking code are much simpler and more straightforward. There is no intermediate UDM involved and the simplified VDM can use post collision velocities and yaw rates directly instead of computing collision paths by integrating collision forces. This means the simplified VDM changes velocity and yaw rate of vehicle directly when it receives a collision packet. This simplifies the integration process, and makes the integration outputs consistent with the results of collision algorithm. It is applicable for simple vehicle dynamics model because the number of control parameters is small, and the correlation among those parameters is low. As a result even abrupt changes of several parameters will not result in system instability.

7. USABILITY TESTING

In order to evaluate the usability of the collision detection and simulation scheme, it is important to consider the system's performance with respect to network delay. This chapter describes the performance of the system under different network delays: how large a network delay is tolerable, what the maximum number of simultaneous simulated vehicles is stable for a given delay, what vehicle velocity is acceptable for collision algorithm under a given delay, etc.

7.1. Usability Analysis

We test usability rather than performance because the principal concern is whether o or not the system as a whole is usable rather than how fast each component can run. It is true that faster component performance can make the system more usable, and sometimes, performance itself is part of usability. But performance without reference is meaningless. In testing the usability of a vehicle collision simulation, the issue that we care most about is whether collisions are successfully detected when they occur, and if they are caught in a timely manner.

In order to give a more specific meaning to these two reference criteria, consider two cases. First, consider the case of an undetected collision. As shown in Figure 7-1, this happens when frame rate of the VDM is so low so that the gap between subsequent packet updates is too large relative to the the velocity of the vehicle. At the n^{th} frame, the vehicle has not reached the barrier. But at the $(n+1)^{th}$ frame , the vehicle has completely passed the barrier. In this case, the collision detection algorithm will fail. If this case occurs when vehicle velocity is within expected limits, then the VDM's update rate is insufficient to support real time vehicle collision simulation. This is more likely the more complex the VDM since this complexity of the model is a major factor in frame time. For testing purpose, the

simplified VDM will be used to make the frame rate as rapid as possible. Different results would be expected for the VDANL method, but the evaluation methods would be similar.

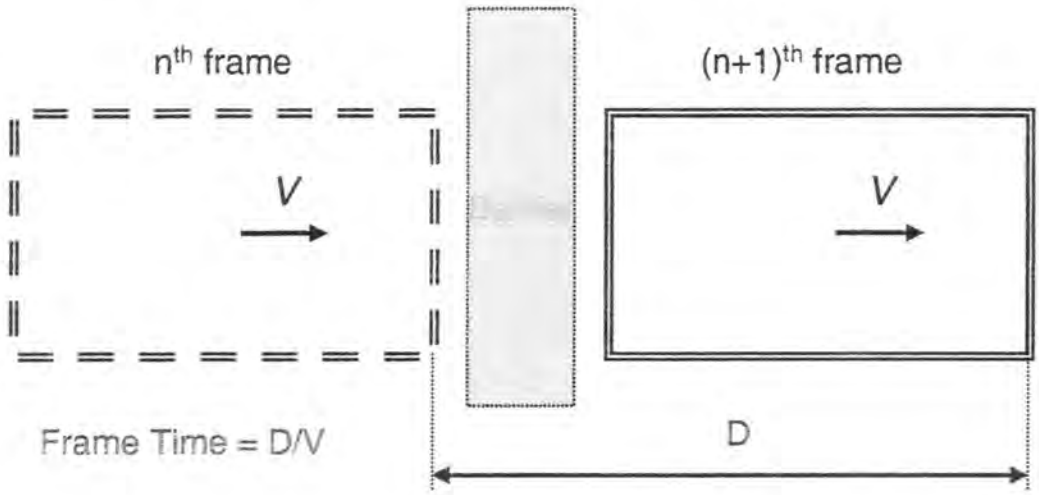

Figure 7-1 Failed Collision Detection

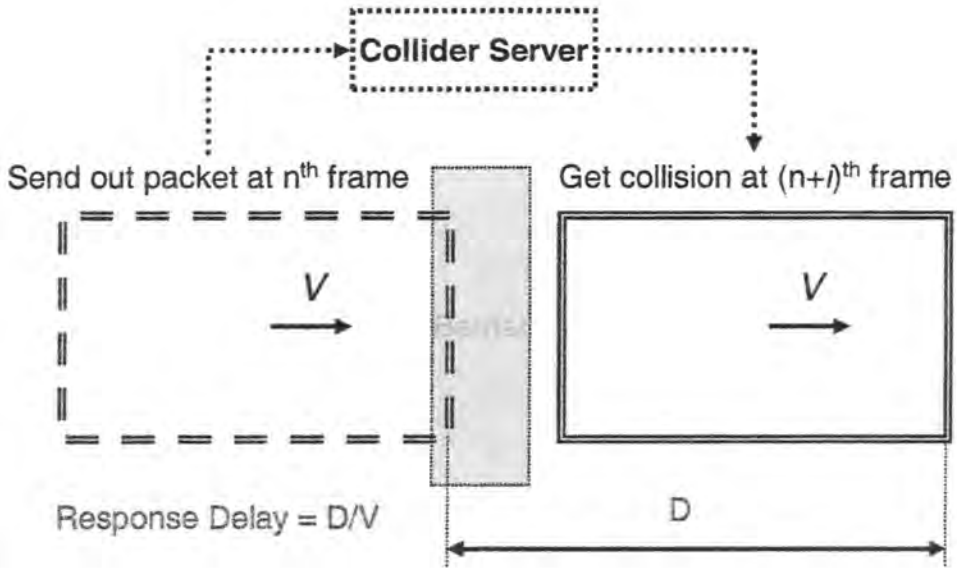

Figure 7-2 Late Collision Detection

In addition to missed collisions, we must also consider collisions which are detected late.. Consider Figure 7-2. A collision occurs at time t_1, when the vehicle collides with the

barrier. But the vehicle does not receive the collision message until time t_2, when it has already passed completely through the barrier. We refer to the elapsed time between when the client sends out a packet to the time when it receives a collision packet back as the response delay. Late collisions occur when the response delay is too large relative to the vehicle velocity.

As shown in Figure 7-2, if V represents vehicle velocity, and Δt_{resp} represents response delay, then the vehicle travel distance due to response delay can be represented as:

$$D = \Delta t_{resp} \times V \qquad (7\text{-}1)$$

In order to maintain D at an acceptable threshold, (say less than a half of vehicle width), the response delay and the maximum vehicle velocity must be limited. Before describing the usability testing, let us first consider more of the details of response delay.

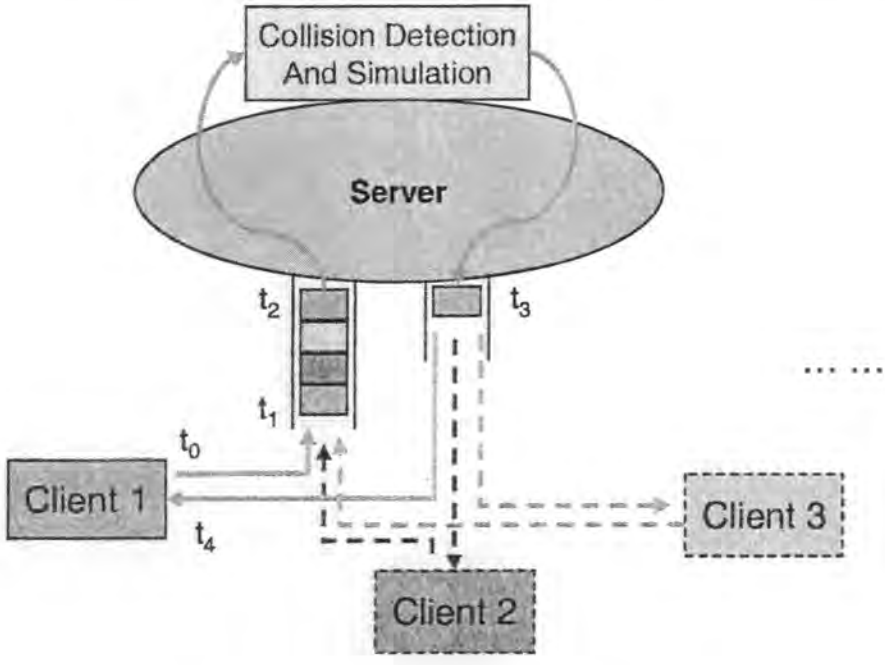

Figure 7-3 Response Delay

Figure 7-3 is a diagram of response delay. The time from t_0 to t_1 is the network delay from client to server. This delay depends solely on network distance and bandwidth. The time

from t_1 to t_2 is manipulation time that depends on number of clients or vehicles (NoV). The time from t_2 to t_3 is collision time, which also depends on the number of vehicles if the collision algorithm is fixed. The time from t_3 to t_4 should be the same as network delay. So response delay consists of network delay, manipulation and collision time. We can merge manipulation time and collision time into a single idea, server time, which is a function of NoV. Assuming no interaction between network delay and server time, the formula for response delay is

$$\Delta t_{resp} = \Delta t_{net} + f(NoV) \qquad (7\text{-}2)$$

Where Δt_{net} is network delay, and NoV is number of vehicles. Merging equation 7-2 and 7-1,

$$D = (\Delta t_{net} + f(NoV)) \times V \qquad (7\text{-}3)$$

Define the reference value for D as half of a vehicle width. If the vehicle travel distance due to response delay is not greater than half of a vehicle's width, collision's can be detected on time. If we conservatively imagine a standard car width as 5 feet, then the condition under which we can guarantee that collisions can be detected on time is:

$$(\Delta t_{net} + f(NoV)) \times V \leq 2.5 \qquad (7\text{-}4)$$

This analysis is, of course, a simplification in that it presumes that network delay is a constant. A more accurate model of network delay is as a stochastic distribution of times categorized by a mean and variance. If the variance of delay is small relative to the mean, then the above analysis holds. For cases where network delay is highly variable, the maximum value for network delay would provide the limit.

7.2. Response Delay vs. Velocity

During vehicle collision simulation, there is a wide range of vehicle velocities we are interested in. Table 7-1 shows the specific speeds that will be studied in this section. The 'G' in first row represents speed for general vehicle.

Table 7-1 Interested Vehicle Speed Range

unit	G1	G2	G3	G_{max}	F-1	Max
Miles/h	60	90	120	150	225	300
Feet/s	88	132	176	220	330	440
Km/h	96	144	192	240	360	480
m/s	26.7	40	53.3	66.7	100	133.4

The relationship between allowed maximum response delay versus vehicle speeds based on formula 7-4 is shown in Figure 7-4 below.

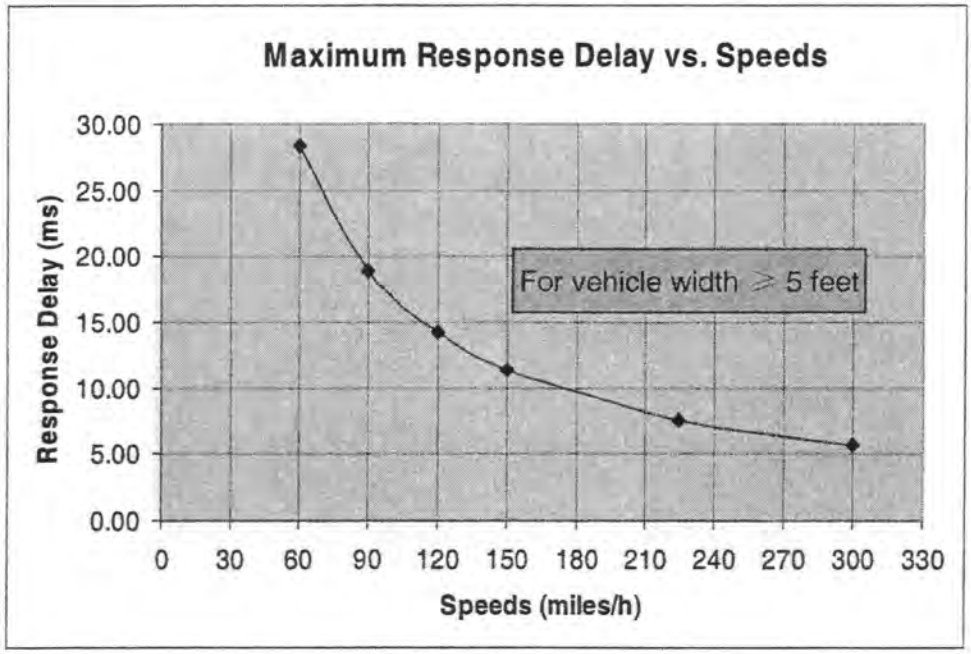

Figure 7-4 Max Response Delay vs. Speeds

As we can see, the maximum response delay for which we can assure on time collision detection at vehicle speed of 150miles/h is 12ms. 150miles/h is a generous maximum speed for all but the most extreme situations involving automobiles.

7.3. Response Delay in LAN

Due to short distance of the 100GB LAN in which the system was tested, the natural network delays were negligibly small relative to the response time limits shown in Figure 7-4. So for our purposes, the network delays can be ignored compared with the server time contribution to response delay. In other words, in testing the system on the dedicated LAN, server time can be treated as equal to response delay. In this section, the relationship between response delay (or server time) in LAN and number of vehicles will be established based on the testing data shown in Table 7-2. Figure 7-5 is the linear regression line of response delay vs. NoV.

Table 7-2 Response Delay in LAN vs. Number of Vehicles

Number of Vehicles	1	2	3	4	5	6	8
Average Res. Delay(ms)	0.4	1.25	1.5	1.75	2.0	2.4	3.0

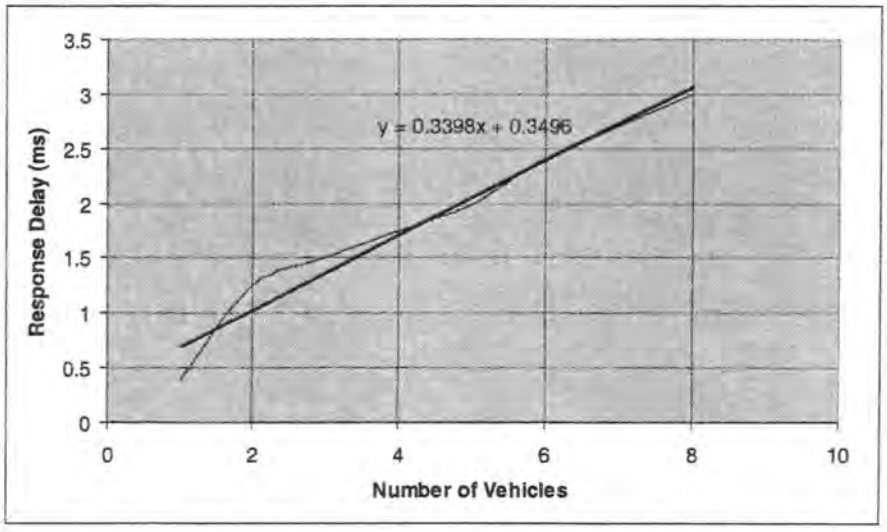

Figure 7-5 Response Delay in LAN vs. Number of Vehicles

Using JMP, we get a linear regression line for average response delay in LAN on NoV. If consider the reasonable maximum velocity of an automobile as 150m/h, this allows a maximum response delay of 12ms. Based on Figure 7-5, we can predict the maximum NoV which can be reliably simulated to be 32 by extrapolating the regression line. The prediction may not be accurate because larger NoV may introduce network traffic that may increase the network delay to significant levels, but it does allow us to say that up to 10 vehicles can be easily simulated on a high-speed LAN with great confidence.

7.4. Response Delay in Other Networks

Table 7-3 Round Trip Time (RTT) for Various Network Types (Preload: 1000 bytes)

Network	LAN	MAN	Inter-city	WAN	Inter-continent
RTT(ms)	<1.0	1-5	5-20	20-100	100-250

Table 7-3 above shows the rough RTT for LAN, MAN, Inter-city network, WAN, and Inter-continental network. Again, if we assume that a vehicles' maximum speed will stay below 150m/h, then the maximum response delay allowed is about 12ms. This limit is well above the expected performance of the LAN and MAN networks. It is possible that even some inter-city networks may satisfy the network delay requirements. However, it is clear that the network architecture used in this thesis would be unsuitable for vehicle collision simulation in WAN or inter-continental networks. Moreover, when the simulation is moved from a LAN environment to a MAN or Inter-city network, the network delay itself becomes a significant contributor to overall response delay. Network delay in MAN or Inter-city is much larger than server time, so under the same simulation condition, the NoV declines dramatically. Table 7-4 and Figure 7-6 show the results of test of the response delay for different NoV in several networks with larger delays.

Table 7-4 Response Delay (ms) vs. NoV and Network Delay

NoV / Network Delay (ms)	1	2	3	4	5	6	8
0	0.4	1.25	1.5	1.75	2.0	2.4	3.0
2	2.5	3.2	3.6	3.8	4.2	4.5	5.0
5	5.4	6.1	6.6	6.9	7.1	7.5	8.1
10	10.6	11.3	11.5	11.7	11.9	12.1	13.0

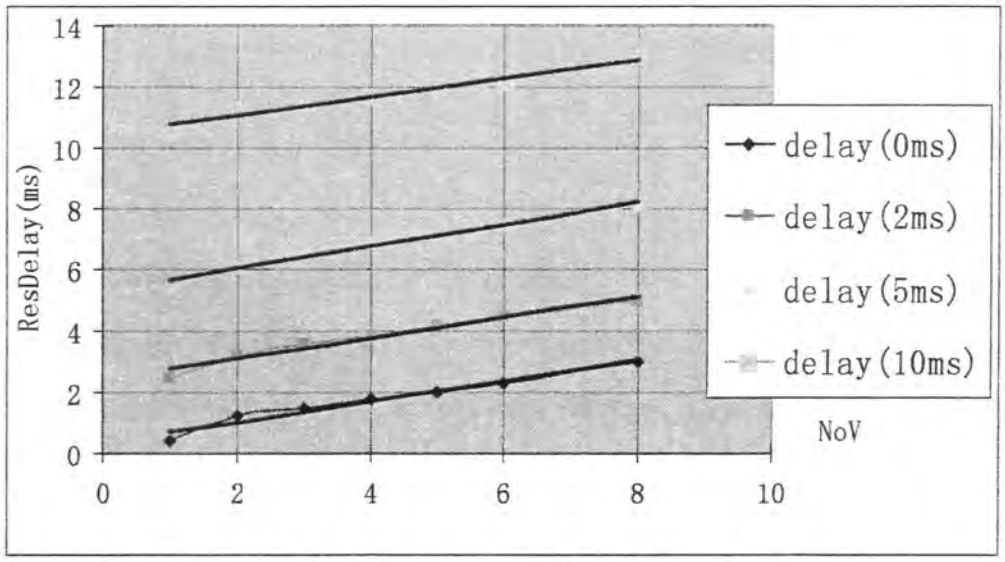

Figure 7-6 Response Delay vs. NoV and Network Delay

8. CONCLUSIONS AND FUTURE WORK

8.1. Conclusions

This thesis presents the design, implementation and evaluation of a network-based vehicle collision detection and simulation system. The system as implemented includes all the components needed for doing a complete vehicle driving simulation. It supports multi-vehicle simulation in multiple places on a network. It supports vehicle-to-scene and vehicle-to-vehicle collision detection and response simulation. Additionally, it supports connection of heterogeneous vehicle dynamics models and image generators.

The OSG built-in hierarchy bounding volume and line segment to surface collision detection functionalities were used for vehicle-to-scene collision detection. This functionality was easily applicable and proved quite efficient for the vehicle collision model used in this thesis. Two stages of collision detection, bounding circle and Cohen-Sutherland clipping, were utilized to facilitate efficient vehicle-to-vehicle collision detection.

A simple, but robust and realistic vehicle-to-vehicle, including vehicle-to-scene, collision simulation algorithm was developed, based on coefficient of restitution presented by Macmillan. A friction model was added to simulate the static friction force. An adapted coefficient of restitution model was created to extend the functionalities of coefficient of restitution (e) so that it could be used for simulating a passing through collision scenario in addition to a receding scenario.

A new collision server was introduced, together with a network architecture to accommodate vehicle-to-vehicle collision detection and response simulation. The network-based collision algorithm presented is capable of reproducing the collision behavior of two colliding vehicles. More importantly, it supplies consistent collision result for each collided vehicle based on the same collision. So vehicle responses in vehicle-to-vehicle collision are consistent, reasonable and realistic.

The flexibility of connection between IG and VDM allows users to choose different connection methods according to different situations. An integrated IG and VDM is the best choice for simple 3D environments with a simple VDM, because the whole system is more compact and easy to operate. However in more complex situations, separating the IG and VDM is a better choice because it allows rendering and vehicle simulation to be computed in parallel. This separation allows the system to more efficiently utilize multiple compute resources to ensure adequate frame rates for both the VDM and the IG.

Test and analysis showed that the network-based vehicle collision detection and simulation system was capable of simulating up to 30 vehicles simultaneously on a dedicated 100Gb LAN with vehicle speeds of up to 150m/h. We also found that the collision simulation with 2-6 vehicles could be done in networks with delays of 10ms, the average delay for an intercity network. These results would be substantially reduced in the presence of network delays where the variance of the delay is significant in comparison to the mean delay.

8.2. Future work

- **To Improve Vehicle Collision Model**

The vehicle collision model used in this thesis was simplified to the four line segments of a rectangle. Thus collisions always happen in the rectangle plane that is vehicle yaw plane at the vehicle cg height. That means any barrier higher or lower than the vehicle's cg height would not generate a collision. This is an obvious flaw of the simplified model. A first order improvement for the model would be to substitute a box for the vehicle body, instead of rectangle, and a circle for each wheel. This would have a small impact on the collision computation, but we expect be significantly more realistic.

- **To Implement 3D Collision Response Simulation**

The simplified vehicle collision model makes several assumptions in computing the collision response. One of them is that the collision responses are applied in the vehicle's yaw

plane at cg height. This assumption may not reasonable for collision cases that lift the vehicle or turn the vehicle over. A true 3D collision response simulation would be more reasonable in these cases than the simplified 2D model used in this thesis. To implement 3D collision response calculation, a 3D vehicle collision model is needed.

- **To Separate Vehicle-to-Scene Collision Calculation from Vehicle-to-Vehicle Collision Calculation**

For simplicity reasons, the vehicle-to-scene and vehicle-to-vehicle collision calculation were both performed on the collision server. Because vehicle-to-scene collision detection can be done on client side, this might be an opportunity to decrease response delay by eliminating network delay.

REFERENCES

1. Johansson, M. and Nordin, J. "A Survey of Driving Simulators and Their Suitability for Testing Volvo Cars," Master Thesis, Chalmers University of Technology Göteborg, Sweden, February 2002.

2. Knight, M.R. "Simulation of Vehicle Collision in Real Time," Master Thesis, Iowa State University, 2002.

3. Gruening, J., Bernard, J., Clover, C., and Hoffmeister, K. "Driving Simulation," SAE Technical Paper Series 980223, 1998.

4. Cremer, J., Kearney, J. and Papelis, Y. "Driving Simulation: Challenges for VR Technology," IEEE Computer Graphics and Applications, pp. 16-20, September 1996.

5. Lee, W.S., Kim, J.H. and Cho, J.H. "Development of a Driving Simulator," IPC-9, Vol. 2, pp. 13-18, November 1997.

6. Romano, R.A., Stoner, J.W., and Evans, D.F. "Real Time Vehicle Dynamics Simulation: Enabling Tool for Fundamental Human Factors Research," SAE Technical Paper Series 910237, 1991.

7. Kallmann, M., Lemoine, P., Thalmann, D., Cordier, F., Magnenat-Thalmann, N., Ruspa, C., and Quattrocolo, S. "Immersive Vehicle Simulators for Prototyping, Training and Ergonomics," Computer Graphics International (CGI), pp. 90-95, July 2003.

8. Balling, O., Knight, M., Walter, B. and Sannier, A. "Collaborative Driving Simulation," SAE Technical Paper Series 2002-01-1222, 2002.

9. Lin, M. and Canny, J. "A fast algorithm for incremental distance calculation," IEEE Conference (Robotics and Automation), pp. 1008-1014, 1991.

10. Cohen, J.D., Lin, M., Manocha, D., and Ponamgi, M.K. "I-COLLIDE: an interactive and exact collision detection system for large-scale environments," Proc. ACM Interactive 3D Graphics Conf., pp. 189-196, 1995.

11. Gottschalk, S., Lin, M. and Manocha, D. "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection," Proc. of ACM Siggraph, 1996.

12. Hudson, T.C., Lin, M., Cohen, J., Gottschalk, S. and Manocha, D. "V-Collide: Accelerated Collision Detection for VRML," Proc. of VRML, 1997.

13. Moore, M. and Wilhelms, J. "Collision Detection and Response for Computer Animation," Computer Graphics, Vol. 22, Num. 3, pp. 289-297, August 1998.

14. Baraff, D. "Fast Contact Force Computation for Non-penetrating Rigid Bodies," Computer Graphics Proceedings, Annual Conference Series, 1994.

15. Baraff, D. "Analytical Methods for Dynamics Simulation of Non-penetrating Rigid Bodies," Computer Graphics, Vol. 23, Num. 3, pp. 223-232, July 1989.

16. Kawachi, K., Suzuki, H. and Kimura, F. "Simulation of Rigid Body Motion with Impulsive Friction Force," Proc. of IEEE International Symposium on Assembly and Task Planning, pp. 182-187, August 1997.

17. Hahn, J.K. "Realistic Animation of Rigid Bodies," Computer Graphics, Vol. 22, Num. 4, pp. 299-308, August 1988.

18. Kamal, M.M. "Analysis and Simulation of Vehicle to Barrier Impact," SAE Technical Paper Series 700414, 1970.

19. Greene, J.E. "Computer Simulation of Car-To-Car Collisions," SAE Technical Paper Series 700015, 1977.

20. "Open Scene Graph Documentation Online Document," http://openscenegraph.sf.net/, visited August 2003.

21. Hill, F.S. Jr. "Computer Graphics Using OpenGL," Upper Saddle River, New Jersey, Prentice Hall, Inc. 2001

22. Macmillian, R.R. "Dynamics of Vehicle Collision, Channel Islands," Jersey, Channel Islands, UK, Inderscience Enterprises Ltd., 1983.

23. Walter, B. "Documentation for the Satellite Server Application," Simulation Sock Documentation, VRAC, Iowa State University, 2001.

24. "VR Juggler Documentation," http://www.vrjuggler.org, visited January 22, 2003.

25. "VDANL User Guide and Technical Reference," Hawthorne, CA, System Technology, Inc. 2003.

26. Gillespie, T.D. "Fundamentals of Vehicle Dynamics," Warrendale, PA: Society of Automotive Engineers, c1992.

27. Mirtich, B. "V-Clip: Fast and Robust Polyhedral Collision Detection," ACM Transactions on Graphics, Vol.17, Issue 3, pp. 177 – 208, July 1998.

28. Schmalstieg, D. and Tobler, R.F. "Real-time Bounding Box Area Computation," Technical Report TR-186-2-99-05, January 1999.

29. Lin, M. and Gottschalk, S. "Collision Detection Between geometric models: a survey," Proceedings of IMA Conference on Mathematics of Surfaces, 1998.