1-1-2002

Simulation of vehicle collisions in real time

Mark Rolland Knight
Iowa State University

Simulation of vehicle collisions in real time

by

Mark Rolland Knight

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 E. Bernard, Major Professor
Adrian Sannier
Frank Chavez

Iowa State University

Ames, Iowa

2002

Graduate College

Iowa State University

This is to certify that the master's thesis of

Mark Rolland Knight

has met the thesis requirements of Iowa State University.

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Jim Bernard, for all his help and guidance throughout my undergraduate and graduate career. I am also very appreciative of my committee members, Dr. Adrian Sannier and Dr. Frank Chavez, for their time and support.

I want to thank all of my colleagues at the Virtual Reality Applications Center. It has been a wonderful environment to work in, and I have learned a great deal from everyone. Thanks to Ole Balling and Systems Technology Inc. for their help with vehicle dynamics issues and specifically the VDANL application.

Finally, I would like to extend my greatest appreciation to my parents, Keith and Nancy, my sister, Karen, and the rest of my wonderful family and friends for their love and support.

ABSTRACT

Typical vehicle simulations require numerical integration at an integration time step no larger than 0.01 seconds, usually less than half that time. This does not leave enough time to carry out the complex calculations required for detailed collision calculations in real time. This thesis presents a method that strikes a compromise, which, although not carrying all the detail necessary for very accurate collision calculations, allows useful simulations to proceed in real time. The method has three parts: collision detection, estimation of the momentum transfer expected to result from the collision, and application of forces to provide the desired momentum transfer. The method uses a common scene graph for collision detection, which allows the system to work with most of the common scene database formats without the need of specialized preprocessing. All of the collision detection and response calculations employ open-source code and are designed to work well at speeds required by real-time vehicle simulation. Examples based on the VDANL vehicle dynamics simulation illustrate the utility of the methodology.

CHAPTER 1: INTRODUCTION

Real-time human-in-the-loop driving simulation is used for a variety of purposes from vehicle design evaluation to driver behavior studies. In real-time vehicle simulations the interaction of the driver with the simulated vehicle and virtual environment is often dependent on how "real" the experience feels. The immersive feeling can be improved through a variety of methods including 3D stereo graphics, sound cues, force feedback steering, and physical motion through a motion base. The addition of collision detection and simulation to human-in-the-loop simulations can improve user immersion and thus the overall driving experience.

The accurate simulation of a vehicle collision typically requires numerically intensive calculations. Nevertheless, collision detection and simulation are not new to either the vehicle dynamics or computer graphics communities. Detailed non-real-time vehicle collision simulations have been in use since the 1960's for applications from assessing vehicle crashworthiness to reconstructing accidents for litigation.

This thesis seeks a compromise between the detailed non-real-time vehicle collision simulations and very simple non-realistic responses. The collision detection and response is simulated through the vehicle dynamics application rather than the image generator. This allows the response to be included in the dynamic behavior of the entire vehicle model, but also requires faster computation times. Quality commercial vehicle dynamics codes require an integration frequency of at least 200 Hz, which is 3 to 20 times faster than typical VR frame rates.

The following chapters present a method to apply collision effects in real time to vehicle dynamics simulations. They describe the collision detection methods and response algorithms, and they present an implementation of the collision library in the context of a commercial vehicle dynamics program, Vehicle Dynamics Analysis Non-Linear (VDANL) from Systems Technology Inc. Finally, the real-time performance is analyzed and possible future work is discussed.

CHAPTER 2: BACKGROUND

Interest in real time collision calculation follows from an interest in driving simulation, which demands real time calculations. References [1,2,3,4] describe several different driving simulators, and reference [5] describes four common components to all driving simulators:

- A simulation of the physics of the vehicle model and the road surface.
- A simulation of the surrounding environment.
- Video and audio displays to display state output to the operator.
- Input control devices for the operators.

Reference [6] further discusses these components of vehicle simulation and adds two additional components for a collaborative driving simulation application, collision interaction and networking management. This thesis focuses on simulating vehicle collisions, wherein there are two key challenges, detecting that a collision has occurred, and computing the effects of the collision in real time.

Detailed collision simulation has been an active area of research since the 1960's. Various lumped mass spring models were presented in the early 1970's [7,8] as a method to evaluate the crashworthiness of vehicles in a more cost effective way. In 1973 McHenry [9] presented the Simulation Model of Automobile Collisions (SMAC) computer program as a tool for accident reconstruction. All of these applications were intended to simulate somewhat detailed collision events, and in the accident reconstruction cases, were often used

in an iterative way to match physical evidence. They were not concerned with real-time performance.

In 1983 Macmillan [10] presented basic rigid body impulse response calculations specifically for vehicle collisions. The method assumes the pre- and post-collision velocities at the impact point are governed by a coefficient of restitution. This method of calculation is appealing for real-time simulation because of its simplicity and speed of the calculations.

Hahn [11] in 1988 presented these same rigid body impulse response calculations for more general rigid bodies in computer animations. About that same time Moore and Wilhelms [12] discussed the topics of collision detection and collision response. They presented two response methods, a spring based penalty method and an impulse based solution. The impulse method was typically faster to compute, especially in violent collisions, and had an added benefit because the resulting system of equations need only be solved once per collision instead of every time step as required by the spring based methods.

This thesis implements the impulse methods given by Macmillan and also presents a method based on the loss of kinetic energy during the collision. Both these response methods give reasonable looking results in real-time. The thesis focuses on collision response calculations, but it will also demonstrate that the real-time performance is highly dependent on the collision detection algorithm speed. Lin [13], Jiménez [14], and Kim [15] have provided recent surveys of collision detection methods.

The following chapters address the issues of real-time collision detection and response and address the challenge of real-time implementation, presenting methods, an application of their solution, and their resulting performance.

CHAPTER 3: COLLISION DETECTION

Typically collisions are detected by searching a database of collidable objects to determine if any interference exists. If a collision is detected, the collision point and collision plane normal are saved to enable calculation of the resulting response. References [11, 12, 13] present several different ways to perform the detection operation.

For this application the Open Scene Graph (OSG) [16] library was selected to perform the collision detection against a visual scene graph database. OSG provides a great deal of flexibility in database formats while still maintaining ample computation speed. In addition, OSG is free, open source, and can run on several computer platforms. Other methods may have a slightly higher speed, particularly with databases containing a high number of polygons, but these methods often require specialized file formats and substantial preprocessing [15].

To test for a collision against the scene graph, the vehicle is represented as a set of line segments that generate a horizontal 2D rectangular plate around the vehicle at the center of gravity (CG) height. For every integration time step, each segment in the bounding rectangle is tested against the scene for intersections. The OSG libraries automate much of this process. The algorithm must simply provide the OSG library with the current list of line segments positioned properly to represent the current location and orientation of the vehicle. The library then traverses the scene graph searching for intersections. The speed of the traversal depends greatly on the spatial organization of the graph. Chapter 6 discusses this in

more detail. Figure 3-1 shows a collision example with the front-right corner of the bounding rectangle intersecting a wall.

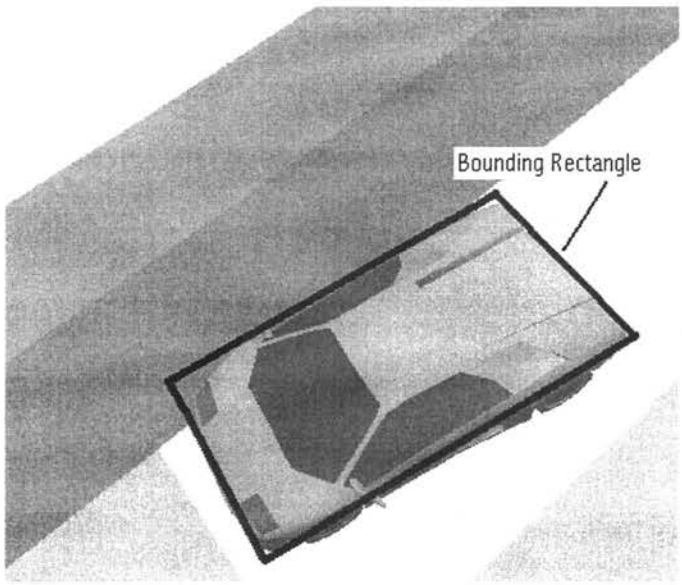

Figure 3-1: Vehicle collision bounding rectangle in a front right side collision.

To enable calculation of the vehicle response to the collision, simple algorithms typically require specification of the force application point and the direction of the force. This method finds the force point of application by computing the midpoint of the line of interference between the two objects. For example, in Figure 3-2 the point of application would be the midpoint of the line segment a-b. Since this application is concerned with flexible vehicle bodies hitting rigid barriers, the collision plane normal is the surface normal of the rigid barrier. The results of the OSG collision traversal provide the collision surface normal and the segment intersection locations used to compute the interference line midpoint.

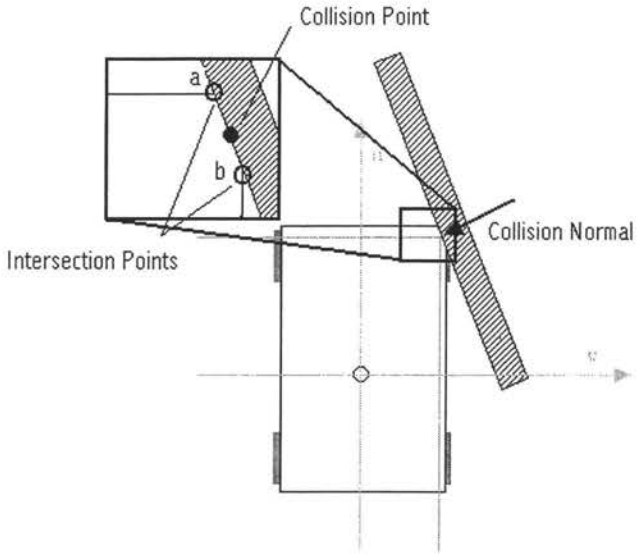

Figure 3-2: Collision application point and collision normal.

CHAPTER 4: COLLISION RESPONSE

The goal of the response is to simulate collisions in a useful way that makes sense but is not intended to be correct in engineering detail. Several simplifying assumptions enable the real-time calculations:

- The impact is between a vehicle and a rigid wall.
- The normal vector to the wall is in the yaw plane of the vehicle.
- The forces of impact are in the yaw plane and at the mass center height of the vehicle.
- The time duration of the impact is small compared to the time scale of yaw plane vehicle motion.
- There is only one impact force, and this impact force remains fixed relative to the vehicle throughout the impact.

This chapter presents two methods for computing the collision response and the resulting force and moment to apply to the vehicle.

4.1 Linear and Angular Momentum Relationships

Both methods presented below begin with the application of momentum relationships. First apply linear momentum:

$$\int_{0}^{t_0+\Delta t} \mathbf{F}dt = m(\mathbf{V}_2 - \mathbf{V}_1) \tag{4-1}$$

where the collision force occurs at time t_0, the duration of the collision is Δt, **F** is the force vector applied to the vehicle by the barrier, m is the mass of the vehicle, $\mathbf{V}_1$ is the yaw plane velocity vector of the vehicle mass center just before the force is applied, and $\mathbf{V}_2$ is the yaw plane velocity vector of the vehicle mass center just after the force is applied. The time interval Δt is assumed to be small enough that other forces, forces from the road on the tires for example, do not affect the momentum transfer. The analysis here is restricted to one impact force. The extension to more impact forces, at least from a momentum point of view, is straightforward.

A similar relationship applies to angular momentum, namely,

$$\boldsymbol{\rho} \times \int_{0}^{t_0+\Delta t} \mathbf{F}dt = I_{zz}(r_2 - r_1) \tag{4-2}$$

where I_{zz} is the vehicle yaw moment of inertia about its mass center, $\boldsymbol{\rho}$ is the vector from the total vehicle mass center to the point of impact, and r_1 and r_2 are the vehicle yaw rates before and after the collision respectively.

Equations (4-1) and (4-2) have a total of three unknowns, the velocity of the mass center after the impact, the angular velocity after the impact, and the impulse. A combination of intuition and experience provides the third equation needed to solve the system. The following sections describe two methods to obtain this additional relationship.

4.2 Coefficient of Restitution Method

First consider the methods presented by Macmillan [10], who calls for the normal velocity change of the vehicle at the point of impact to be a function of a coefficient of restitution.

$$\mathbf{Vp_2} \cdot \mathbf{N} = -e\,(\mathbf{Vp_1} \cdot \mathbf{N}) \tag{4-3}$$

$\mathbf{Vp_1}$ and $\mathbf{Vp_2}$ are the velocity vectors of the vehicle at the point of impact before and after the collision and $\mathbf{N}$ is the unit normal to the rigid surface. The so-called coefficient of restitution e is a function of the details of the collision. Macmillan suggests values in the range of 0.0 to 0.3 for most vehicle collisions. Equations (4-1), (4-2), and (4-3) yield the impulse, post collision angular velocity, and velocity of the mass center.

The algorithm implementation defines the collision frame of reference about the point of collision with a primary force along the surface normal of the collision and a frictional force along the tangent direction. Macmillan's presentation can be applied to the general two-vehicle collision, but the presentation here, which uses Macmillan's nomenclature, is limited to the special case of a single vehicle hitting a static rigid body.

Given the vehicle velocity vector projected into the collision normal and tangential directions, the following momentum equations relate the pre- and post-collision velocities with the impulse:

$$m\cdot(vn_2 - vn_1) = \text{Impulse} \tag{4-4}$$

$$m\cdot(vt_2 - vt_1) = \mu\cdot\text{Impulse} \tag{4-5}$$

Where vn is the mass center velocity normal to the collision surface, vt is the mass center velocity tangent to the collision surface, μ is the coefficient of friction, and Impulse is the impulse from Fn, normal to the collision surface.

$$\text{Impulse} = \int_{t_0}^{t_0+\Delta t} F_n \, dt \tag{4-6}$$

The change in angular momentum is given by,

$$I_{zz}\cdot(r_2 - r_1) = \text{Impulse}\cdot(\mu\cdot x - y) \tag{4-7}$$

where r is the yaw rate and x and y are the distances from the total vehicle mass center to the point of collision as illustrated in Figure 4-1.

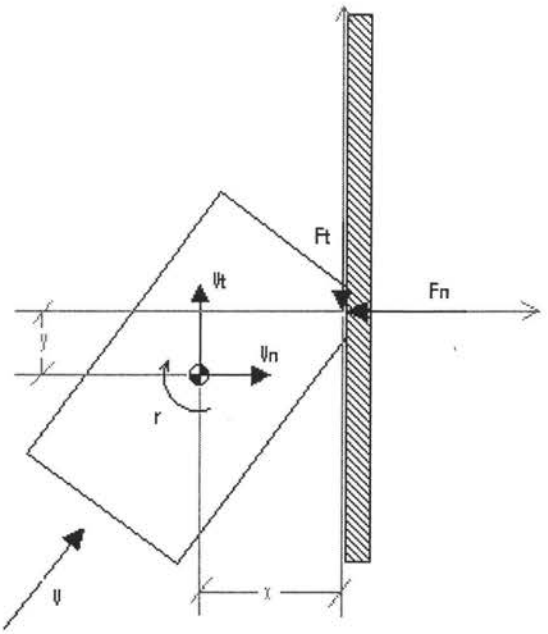

Figure 4-1: Restitution method collision diagram.

From Equation (4-3) the pre- and post-collision velocities are related by

$$p_2 = -e \cdot p_1 \tag{4-8}$$

where p_1 and p_2 are the vehicle velocities at the point of impact and normal to the collision surface and e is the coefficient of restitution. It is clear from Figure 4-1 that

$$p_1 = vn_1 - r_1 \cdot y \tag{4-9}$$

$$p_2 = vn_2 - r_2 \cdot y \tag{4-10}$$

where y is the moment arm from the vehicle's mass center to the collision normal.

The coefficient of restitution specifies how much velocity and thus energy is retained in the system. According to Macmillan, typical vehicle collisions have coefficients of restitution of between 0.0 and 0.3. Based on an interpretation of the results for a range of collisions simulated in real time, it is useful to set the coefficient of restitution as a function of the angle of attack between the vehicle and the wall, from 0.3 for small angles to 0.05 at a 90-degree impact. Figure 4-2 shows a cosine function of the angle of attack that fits the desired relationship well. The function is defined as:

$$e(\alpha) = 0.125 \cdot \cos(2 \cdot \alpha) + 0.175 \qquad (4\text{-}11)$$

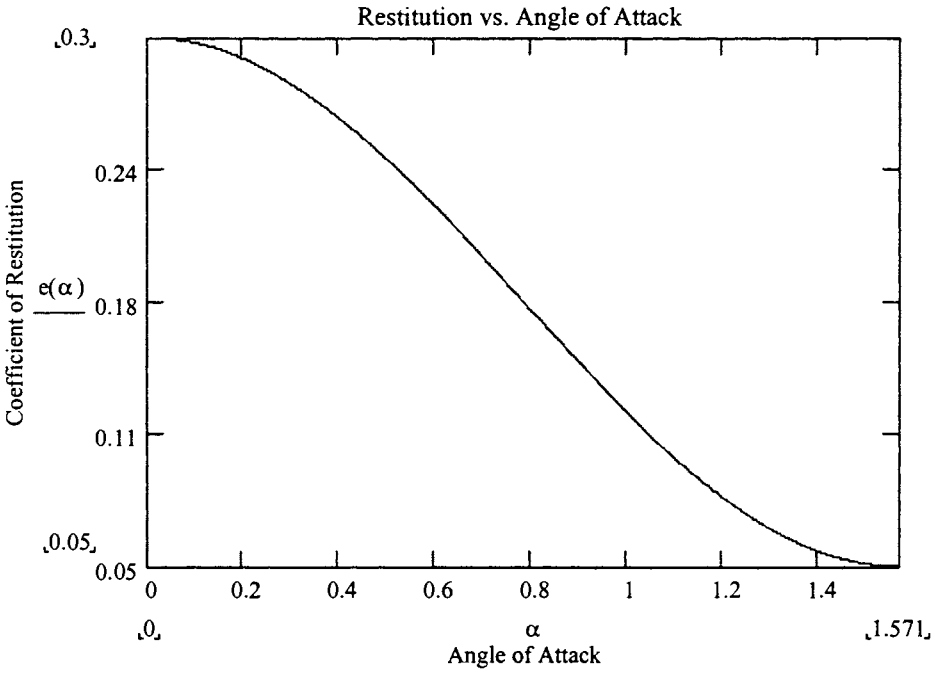

Figure 4-2: Coefficient of restitution vs. angle of attack.

The coefficient of restitution relationship and the momentum equations yield the impulse. Substituting Equation (4-10) into Equation (4-8) yields:

$$vn_2 - r_2 \cdot y = -e \cdot p_1 \tag{4-12}$$

where, p_1 is computed from Equation (4-9) and the pre-collision velocities. Using the momentum equations (4-4 through 4-7), rearranging and substituting into Equation (4-12) yields

$$\frac{m \cdot vn_1 + Impulse}{m} - y \cdot \left[r_1 + \frac{Impulse(\mu \cdot x - y)}{I_{zz}} \right] = -e \cdot p_1 \tag{4-13}$$

Simplifying,

$$p_1 + Impulse \cdot \left[\left(\frac{1}{m} + \frac{y^2}{I_{zz}} \right) - \mu \left(\frac{x \cdot y}{I_{zz}} \right) \right] = -e \cdot p_1 \tag{4-14}$$

Now, if,

$$a = \frac{1}{m} + \frac{y^2}{I_{zz}} \tag{4-15}$$

and,

$$b = \frac{x \cdot y}{I_{zz}} \tag{4-16}$$

then the impulse is

$$\text{Impulse} = -\frac{1+e}{a-\mu\cdot b}\cdot p_1 \tag{4-17}$$

There are two ways to apply this impulse in the context of the simulation: One method is to use the impulse, initial velocities, and momentum equations to compute the post collision linear and angular velocities. The new velocities then replace the vehicle velocities and the integration is continued along the new path of travel. This is the typical application of an impulse treated as an instantaneous event. This method is not practical for this particular application because of the interaction with third-party commercial dynamics software, which cannot reset state variables and restart the integration quickly enough to meet the real-time constraints.

Another method of application is through the use of a collision force and moment. Assuming the impulse can be applied over a small time step, one can compute a constant force to apply over that time step to achieve the desired impulse. Using this method the following forces can be applied to the vehicle:

$$Fn = \frac{\text{Impulse}}{\Delta t} \tag{4-18}$$

$$Ft = \frac{\mu\cdot\text{Impulse}}{\Delta t} \tag{4-19}$$

where Fn and Ft denote the force along the collision normal and tangent respectively. Finally the moment is computed from the collision force and vehicle geometry.

$$\textbf{Moment} = \rho \times (\textbf{Fn} + \textbf{Ft}) \tag{4-20}$$

where ρ is the moment arm from the point of impact to the vehicle mass center.

4.3 Kinetic Energy Loss Method

Another method to acquire the additional information to solve for the collision force and moment is to stipulate the desired energy loss during the collision. In particular, consider the parameter P, which indicates the fraction of yaw plane energy remaining after the collision.

$$P = \frac{m(\mathbf{V}_2 \cdot \mathbf{V}_2) + I_{zz} \cdot r_2^{\,2}}{m(\mathbf{V}_1 \cdot \mathbf{V}_1) + I_{zz} \cdot r_1^{\,2}} \tag{4-21}$$

As in the coefficient of restitution method, it is useful to make P a function of the angle of attack between the vehicle and a rigid wall. Again, a cosine function works well. Figure 4-3 presents a plot of the function we have found useful.

$$P(\alpha) := 0.44\cos(2\cdot\alpha) + 0.4$$ (4-22)

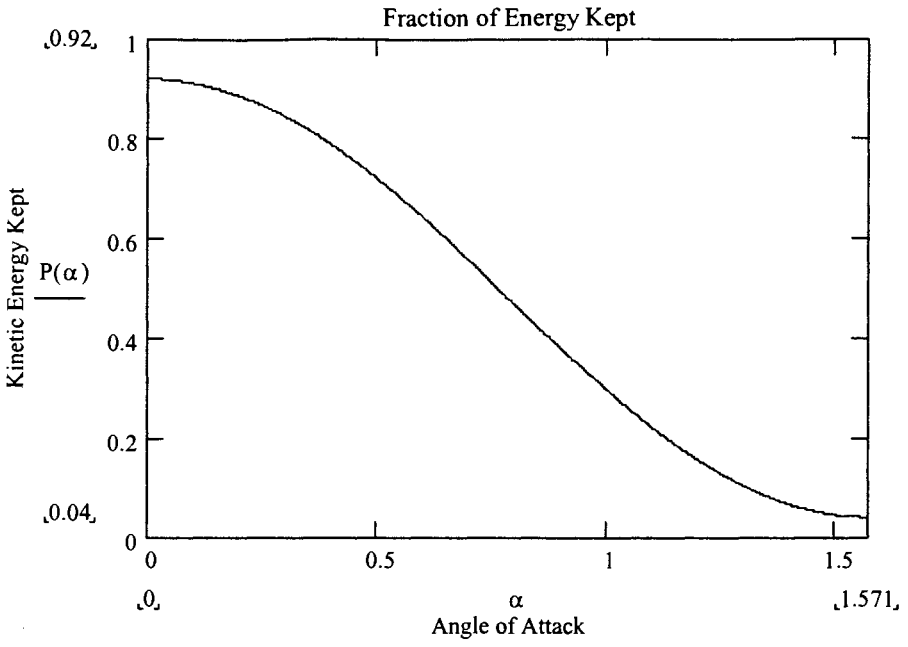

Figure 4-3: Fraction of kinetic energy kept as a function of collision angle of attack.

The collision force is

$$\vec{F} = F_0(a\mathbf{i} + b\mathbf{j}) \tag{4-23}$$

where **i** and **j** are unit vectors in the vehicle's local coordinate system (see Figure 4-4). F_0 is the magnitude of the force.

Following the procedure of the restitution method, linear momentum is applied for the system. Referring to Figure 4-4 and assuming a small finite time Δt, Equation (4-1) can be written in scalar form

$$F_0 \cdot a = \frac{m \cdot (u_2 - u_1)}{\Delta t} \tag{4-24}$$

$$F_0 \cdot b = \frac{m \cdot (v_2 - v_1)}{\Delta t} \tag{4-25}$$

where u and v are the longitudinal and lateral components of the mass center velocity.

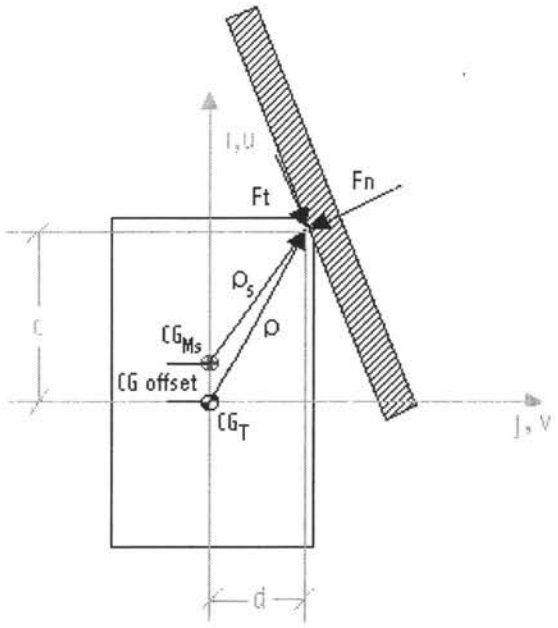

Figure 4-4: Kinetic energy method force diagram for a front right corner collision.

Similarly, the change in angular momentum about the mass center is given by Equation (4-2) and can be computed in scalar form as:

$$F_0 \cdot (a \cdot -d + b \cdot c) = \frac{I_{zz} \cdot (r_2 - r_1)}{\Delta t} \tag{4-26}$$

where r is the yaw rate, I_{zz} is the yaw moment of inertia, and, c and d are illustrated in Figure 4-4.

$$\rho = (c\mathbf{i} + d\mathbf{j}) \tag{4-27}$$

The relationship between pre- and post-collision kinetic energy is given by

$$E_2 = P \cdot E_1 \tag{4-28}$$

where P is the fraction of kinetic energy maintained after the collision and

$$E_1 = 0.5 \cdot m \cdot \left(u_1^2 + v_1^2 \right) + 0.5 \cdot I_{zz} \cdot r_1^2 \tag{4-29}$$

$$E_2 = 0.5 \cdot m \cdot \left(u_2^2 + v_2^2 \right) + 0.5 \cdot I_{zz} \cdot r_2^2 \tag{4-30}$$

These equations yield the force magnitude F_0 required to achieve the impulse across the specified collision time step Δt. First, the initial conditions are used to compute the initial energy E_1 with Equation (4-29). The final energy E_2 is then computed with Equation (4-28), the initial energy, and the known fraction of energy maintained across the collision.

From the momentum equations relate the post-collision velocity and yaw rate to the magnitude of the impact force:

$$u_2 = \frac{F_0 \cdot a \cdot \Delta t + m \cdot u_1}{m} \tag{4-31}$$

$$v_2 = \frac{F_0 \cdot b \cdot \Delta t + m \cdot v_1}{m} \tag{4-32}$$

$$r_2 = \frac{\left(F_0 \cdot \Delta t \cdot b \cdot c - F_0 \cdot \Delta t \cdot a \cdot d + I_{zz} \cdot r_1\right)}{I_{zz}} \tag{4-33}$$

Now Equation (4-30) yields,

$$\frac{E_2}{0.5} = \frac{m \cdot \left(F_0^2 \cdot \Delta t^2 \cdot a^2 + 2 \cdot F_0 \cdot \Delta t \cdot a \cdot m \cdot u_1 + m^2 \cdot u_1^2\right)}{m^2} + \ldots.$$

$$.. + \frac{m \cdot \left(F_0^2 \cdot b^2 \cdot \Delta t^2 + 2 \cdot F_0 \cdot b \cdot \Delta t \cdot m \cdot v_1 + m^2 \cdot v_1^2\right)}{m^2} + I_{zz} \cdot r_2^2 \tag{4-34}$$

which can be rewritten as

$$\frac{E_2}{0.5} = F_0^2 \cdot \left[\frac{\Delta t^2 \cdot \left(a^2 + b^2\right)}{m} + \frac{\Delta t^2 \cdot \left(a^2 \cdot d^2 + b^2 \cdot c^2 - 2 \cdot a \cdot d \cdot b \cdot c\right)}{I_{zz}}\right] + \ldots.$$

$$\ldots. + F_0 \cdot \left[2 \cdot \Delta t \cdot \left(a \cdot u_1 + b \cdot v_1\right) + 2 \cdot \Delta t \cdot r_1 \cdot (b \cdot c - a \cdot d)\right] + \ldots.$$

$$\ldots. + m \cdot \left(u_1^2 + v_1^2\right) + I_{zz} \cdot r_1^2 \tag{4-35}$$

The final solution for F_0 is

$$F_0 = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{4-36}$$

Where:

$$A = \frac{\Delta t^2 \cdot \left(a^2 + b^2\right)}{m} + \frac{\Delta t^2 \cdot \left(a^2 \cdot d^2 + b^2 \cdot c^2 - 2 \cdot a \cdot d \cdot b \cdot c\right)}{I_{zz}} \tag{4-37}$$

$$B = 2 \cdot \Delta t \cdot \left(a \cdot u_1 + b \cdot v_1\right) + 2 \cdot \Delta t \cdot r_1 \cdot (b \cdot c - a \cdot d) \tag{4-38}$$

$$C = m \cdot \left(u_1^2 + v_1^2\right) + I_{zz} \cdot r_1^2 - \frac{E_2}{0.5} \tag{4-39}$$

Implementation of this algorithm has indicated that the larger of the two forces prevents the vehicle from going through the barrier, and the smaller allows the vehicle to go through the barrier. Although the larger force is usually used, it is clear that the lower force may also be useful – say for a head on crash through a barrier, which causes the energy loss given by the parameter P.

The plot in Figure 4-5 illustrates the two solutions for this method. The plot shows an impact with a collision point in the front center of the vehicle. It illustrates the effect of the angle of attack on energy loss capability and the resulting force magnitude. Notice the 90-degree case curve covers the entire range from 100%-0% energy maintained. At the 100% energy condition there are two possible solutions. One has the force magnitude equal to zero and thus the vehicle remains moving forward at its current velocity. The other solution applies a large backward force such that the vehicle maintains 100% kinetic energy, but is

moving backward at the initial speed. The other curves show the results with differing angles of attack. As the angle of attack decreases, the maximum possible energy loss drops off because the normal force is decreasing and there is a limit, μ*Fn, for how large the frictional force can get to achieve the desired energy loss. It is also interesting to see the 90-degree coefficient of restitution collision result superimposed on the energy solution. It matches the larger, more negative, force solution from the kinetic energy method, which does not allow the vehicle through the wall. For the purposes of the collision library we apply the larger force solution that does not allow the vehicle to penetrate the wall.

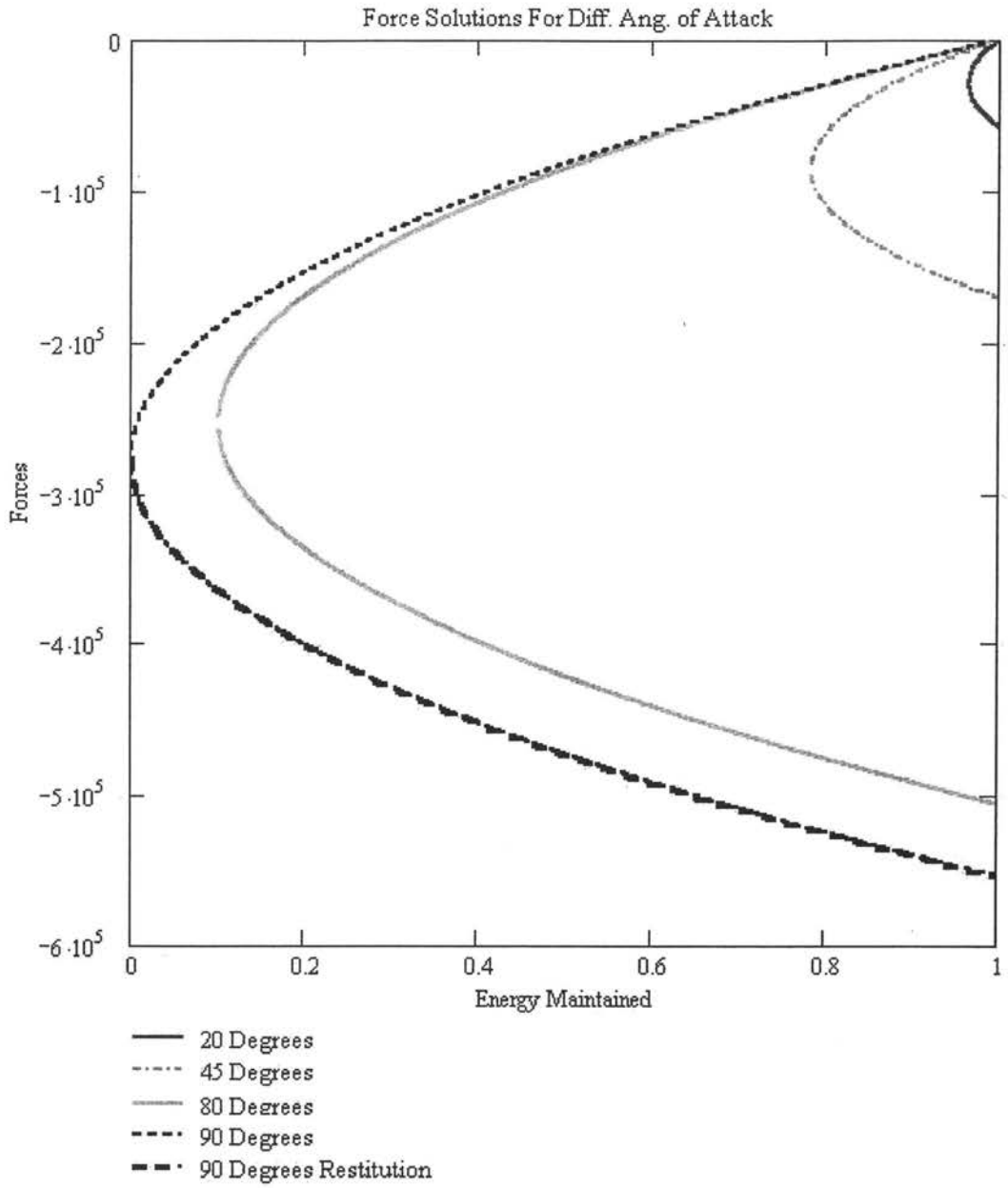

Figure 4-5: Force magnitude vs. kinetic energy maintained for varying angles of attack.

CHAPTER 5: VDANL IMPLEMENTATION

The methods presented here can be used with any vehicle dynamics platform. There are two choices for implementation upon collision detection and calculation of the momentum change:

1. Reset the state variables of the simulation with the post collision velocity and yaw rate and continue integration.
2. Apply a constant force over a short period of time, which will cause the desired momentum change.

The first method, resetting the state variables, is most suitable for modelers who have control of the vehicle dynamics software. It is not very practical for modelers using commercial code because in the context of such code the modeler does not typically have the ability to overwrite the state variables and restart the motion quickly enough to meet the real-time constraints. This thesis uses the second method, namely, applying the force required to cause the desired momentum change.

For this application, both the coefficient of restitution method and the energy method were implemented with the commercial software VDANL [17]. The implementation assumed a constant force between the rigid barrier and the vehicle for a small period of time to yield the desired momentum change, and used VDANL's User Defined Module option to apply the collision forces and moments to the vehicle model.

For both response methods, the force of impact was composed of a force normal to the surface of collision and a frictional force tangent to the surface. A coefficient of friction μ was used to define the friction force F_t as

$$F_t = \mu \cdot F_n \tag{5-1}$$

Macmillan suggests coefficient of friction values from 0.0-0.3 for typical vehicle collisions. We have found it useful to make the coefficient of friction a function of the angle of attack from 0.3 at small angles of attack to 0.0 for head-on collisions where we expect the direction of the impact force to be normal to the collision surface.

To implement the method, the collision scene and vehicle parameters are initialized before the simulation begins. Then, at the initiation of each integration time step, the algorithm tests for collisions. If a collision is detected, the resulting force and moment are computed and applied to the vehicle dynamics model.

5.1 Initialization

Prior to starting the integration, the user must initialize the collision scene graph and supply vehicle parameters for the collision testing and response calculations. The collision scene file can be any visual database format supported by Open Scene Graph [16]. The Virtual Reality Applications Center primarily uses the OpenFlight format. The scene graph libraries internally handle the database initialization from a specified database file. The rest of the initialization deals with setting up the vehicle representation and parameters.

The following parameters are required:

- Vehicle mass
- Vehicle yaw moment of inertia
- Distance from the center of gravity (CG) to the front of the vehicle body
- Distance from the CG to the rear of the vehicle body
- Vehicle body width
- CG height

The various dimensions are used to generate a 2D bounding rectangle, which is set at the CG height.

5.2 Collision Testing and Force Computation

For each dynamics time step the simulation must supply the collision algorithm with the simulation time, linear position and velocity, and angular position and velocity of the vehicle. This allows the library to move the bounding rectangle in the scene and test for interference with any of the scene's elements. If a collision is detected, the collision point and collision surface normal are stored and the force and moment are computed using the methods in Chapter 4 for application to the sprung mass.

5.3 Application of Force and Moment

Equations (4-1) and (4-2) plus equation (4-3) or (4-4) enable the calculation of the impulse $\int \mathbf{F} dt$ and the angular impulse $\boldsymbol{\rho} \times \int \mathbf{F} dt$. We have implemented these impulses in the context of VDANL by assuming a constant force **F** applied over a small time step to achieve the momentum change. Thus,

$$\int_{t_o}^{t_o+\Delta t} \mathbf{F} dt = \mathbf{F} \cdot \Delta t \qquad (5\text{-}2)$$

We verified that the fixed step integrator of VDANL provides accurate results with a collision duration Δt of one integration time step,

$$\Delta t = 0.005 \quad \text{second} \qquad (5\text{-}3)$$

CHAPTER 6: PERFORMANCE

The performance testing focused on the two primary objectives, that the results look reasonable and follow rigid body momentum and energy relationships correctly, and that the algorithms compute in real time.

The momentum and energy calculations were checked using two different operating scenarios (see Figures 6-3 and 6-4). One scenario examined a head-on collision at approximately 30 mph and the other scenario examined a front right side impact with a 10-degree angle of attack at 60 mph. Each scenario was run with the two collision response methods. Comparing the pre- and post-collision velocity vectors and yaw rates verified that the algorithms run correctly.

The real-time performance of the algorithm was tested by using two databases, one simple and one more complex. The complexity of a database is primarily measured by polygon count. Figure 6-1 presents the simple database, a test scene containing an L-shaped wall and a flat driving surface. The database contains 612 triangles, all of which are used to test for collisions. Figure 6-2 presents part of the more complicated Watkins Glen racetrack scene. The entire visual scene consists of 8780 triangles, but we created a version of the database with only the vertical elements for collision testing. This removed unnecessary terrain and sky polygons and brought the collision triangle count to 1838.

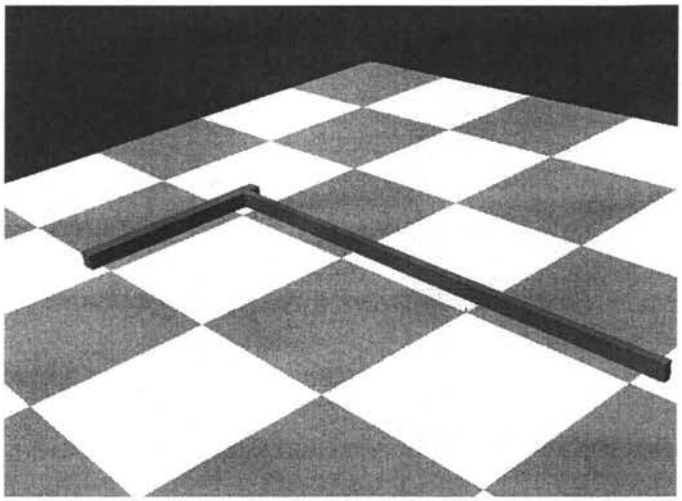

Figure 6-1: L-shaped test scene.

Figure 6-2: Watkins Glen track.

Numerical experiments verified the expectation that the collision computation speed is mainly dependent on collision scene complexity. All of the experiments were run on a 300 Mhz Pentium II PC with 192 MB of RAM. Both the restitution and kinetic energy based methods yield fast computation times for the collision response portion of the algorithm

averaging 0.013 ms per integration time step independent of the complexity of the database. This shows the response calculation time is negligible as it is only 0.26% of the dynamics integration time step of 5 ms.

The collision detection calculations can take a large portion of the 5 ms time step. Table 6-1 presents the average dynamics update times for the numerical experiments. These update times include the normal vehicle dynamics calculations as well as all collision detection and response calculations. Since the dynamics and collision response calculation speeds are relatively constant, this table presents a good measure of relative detection speeds for different scenes. The table illustrates how the more complicated scene requires higher query times and thus longer update times.

Table 6-1: VDANL update method times using different collision scenes.

	Kinetic Energy Method	Coefficient of Restitution Method
Test Rail	2.52 ms	2.60 ms
Watkins Glen	4.69 ms	4.64 ms
Watkins Glen Optimized by MultiGen Creator	3.88 ms	3.90 ms
No Collision Calculation	0.99 ms	

Both scenes were created with Multigen Creator, which can optimize the scene graph hierarchy to improve the collision detection speed [18]. Table 6-1 shows update times for both models tested in a non-optimized state and also shows the Watkins Glen database after the Creator optimization. This optimization reorganizes the polygons spatially so collisions are detected more quickly by only querying polygons in close proximity to the vehicle. Optimizing the Watkins Glen database improved the overall performance by over 15%.

These data indicate that, depending on what other activities need to be completed during the time step (i.e. terrain queries, networking, etc.), the simulation can begin to have computation times outside of real-time if the user is not careful to manage the various computation speeds.

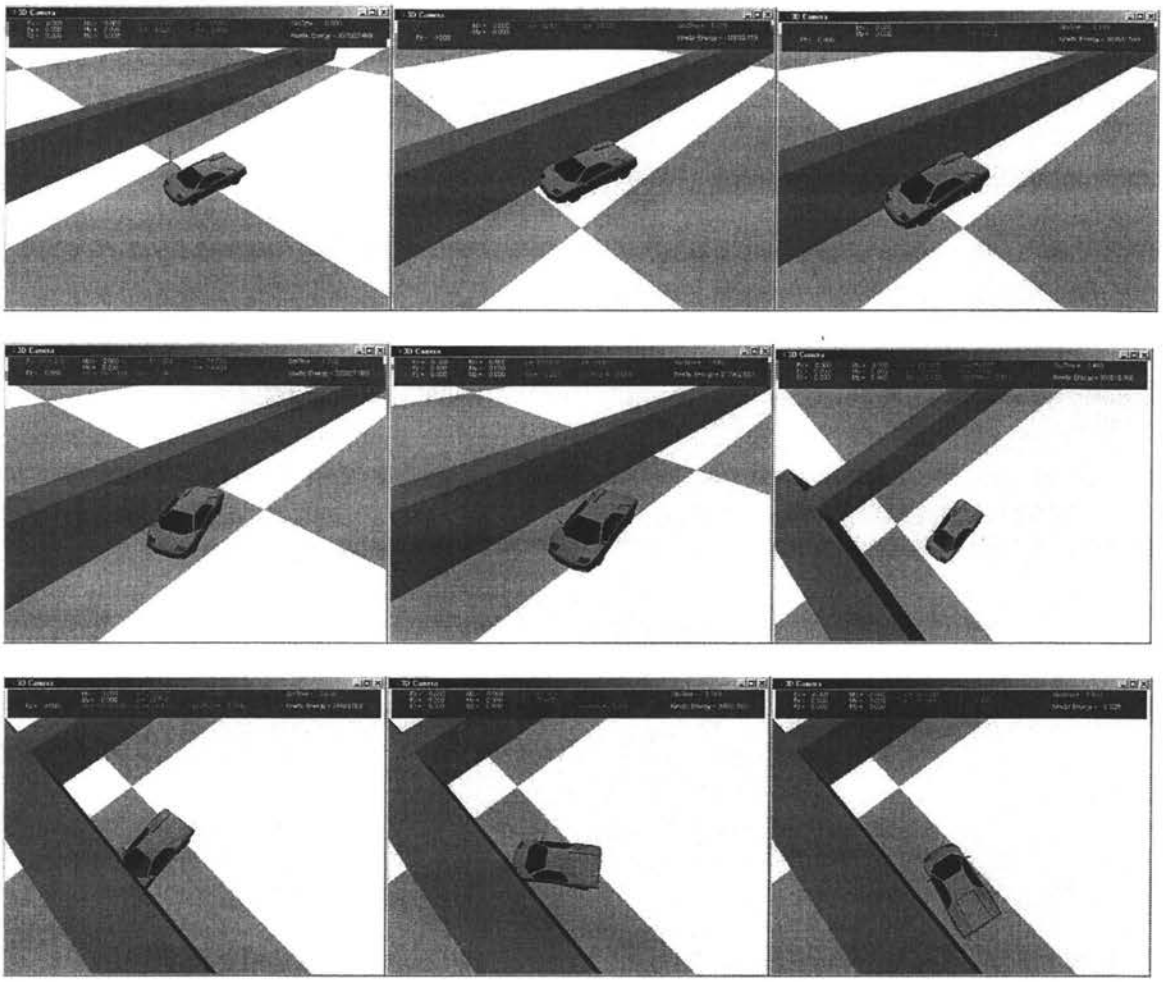

Figure 6-3: Animation stills for the front right side hit collision test scenario.

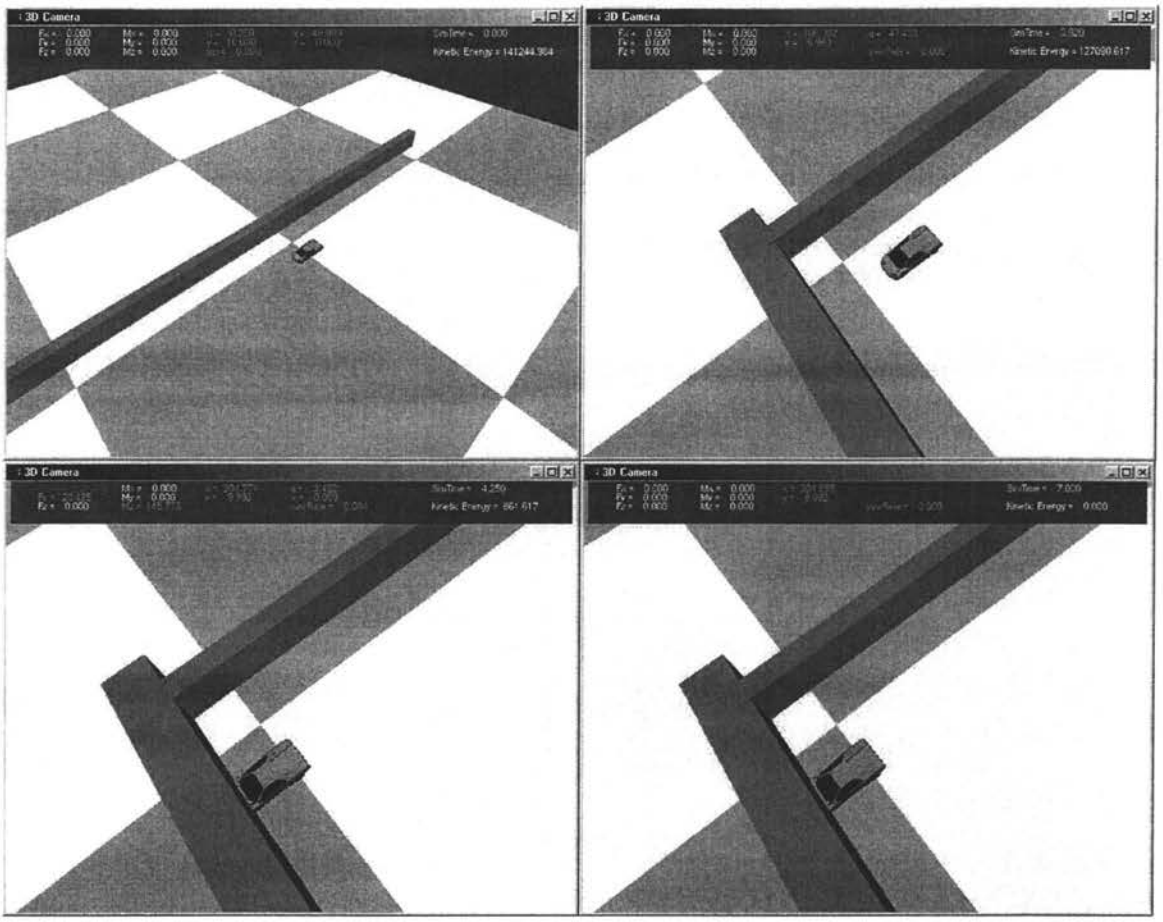

Figure 6-4: Animation stills for the head on collision test scenario.

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

The topic of collision simulation is not new in the vehicle dynamics and computer graphics communities. Detailed collision simulations are often used by the vehicle design community to evaluate the vehicle structure and in litigation to reconstruct accidents. Many VR applications have some sort of collision detection, each with varying levels of complexity and accuracy.

This thesis presented an algorithm that enables real-time collision simulation for human-in-the-loop driving simulations. The impulse momentum methods strike a compromise between highly detailed vehicle collision simulations and very simple reactions sometimes used in virtual reality. It discussed the three steps to compute a collision: collision detection, computation of the resulting impulse, and application of the force and moment to a vehicle dynamics model. The VDANL based implementation illustrated real-time collision in simple and moderately complex databases. The resulting collision library was composed completely of open source code and can be implemented in almost any real-time vehicle simulation.

In the future, the library could be extended to include collisions with moving objects. The primary interest is in vehicle-to-vehicle collisions for collaborative driving simulations. Modification of this code to allow for moving collidable bodies will be challenging, particularly when implemented for vehicles simulated on different dynamics engines but interacting in the same environment. Continued improvement in computation speed will help in the extension of this work to include additional vehicles and more complicated databases.

REFERENCES

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

2. Greenberg, J.A., Park, T.J., "The Ford Driving Simulator", SAE Paper 940176, 1994.

3. Bertollini, G.P., et al., "The General Motors Driving Simulator", SAE Paper 940179, 1994.

4. Chen, L.D., Papelis, Y., Watson, G., Solis, D., "NADS at the University of Iowa: A Tool for Driving Safety Research", Paper presented at the 1st Human-Centered Transportation Simulation Conference, Iowa City, IA, 2001.

5. Gruening, J., Bernard, J., Clover, C., Hoffmeister, K. "Driving Simulation", SAE Paper 980223, 1998.

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

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

8. Greene, J.E., "Computer Simulation of Car-To-Car Collisions", SAE Paper 770015, 1977.

9. McHenry, R.R., "Computer Program for Reconstruction of Highway Accidents", SAE Paper 730980, 1973.

10. Macmillian, R.H., Dynamics of Vehicle Collisions, Channel Islands, UK: Inderscience Enterprises Ltd., 1983.

11. Hahn, J.K., "Realistic Animation of Rigid Bodies", Computer Graphics, Volume 22, Number 4, 1988.

12. Moore, M. and Wilhelms, J., "Collision Detection and Response for Computer Animation", Computer Graphics, Volume 22, Number 4, 1988.

13. Lin, C.M., Gottschalk, S., "Collision Detection Between Geometric Models: A Survey", University of North Carolina, 1998.

14. Jiménez, P., Thomas, F., and Torras C., "3D Collision Detection: A Survey", Institut de Robòtica i Informàtica Industrial, Barcelona, Spain, 2000.

15. Kim, C., "Collision Detection Algorithms", Virtual Reality Applications Center, Iowa State University, 2002.

16. Open Scene Graph Documentation, Online. www.openscenegraph.org. Date retrieved: November 12, 2002.

17. Allen, W.R., Rosenthal, T.J., Klyde, D.H., Chrstos, J.P., "Vehicle and Tire Modeling for Dynamic Analysis and Real-Time Simulation", SAE Paper 2000-01-1620.

18. MultiGen Creator Users Guide, MultiGen-Paradigm Inc., 1999.