In *The Journal of Visualization and Computer Animation*, Vol. 4, No. 1, 1993, pp. 13-24.

NPSNET: REAL-TIME VEHICLE COLLISIONS, EXPLOSIONS AND TERRAIN MODIFICATIONS

Michael J. Zyda*, William D. Osborne, James G. Monahan, David R. Pratt
Naval Postgraduate School
Department of Computer Science
Monterey, California 93943-5100 USA
Email: zyda@trouble.cs.nps.navy.mil

Abstract

The Graphics and Video Laboratory of the Naval Postgraduate School (NPS) is in the process of constructing a three dimensional (3D) virtual world called NPSNET [Zyda91]. NPSNET is a low-cost, commercial workstation-based 3D visual simulator that utilizes SIMNET terrain databases and networking [Garv88]. NPSNET is programmed utilizing off-the-shelf SGI IRIS graphics workstation, rather than the platform specific nodes of SIMNET. Part of the work in constructing the NPSNET world is detecting and resolving collisions in real-time. Such collision detection and resolution has been accomplished and integrated into the latest version of NPSNET, NPSNET-2. The detection of vehicle-weapon, vehicle-vehicle and vehicle-stationary nonvehicle collisions is performed throughout the virtual world.

Background

Users of NPSNET are able to drive many different types of vehicles such as tanks, jets, ships, helicopters and armored personnel carriers. Up to 500 of the vehicles/objects can maneuver in the world at one time. The 500 vehicles can include autonomous vehicles that react when fired upon by either returning fire or fleeing the area. These 3D icons move around in the computer world that is based on the terrain at Fort Hunter-Liggett, California. The terrain appearance is enhanced by the inclusion of textures on many terrain details. Depending upon the model of IRIS being used, the user can select to use texturing, fog and even haze. The system is networked, via Ethernet, to allow several players to interact. A two dimensional (2D) map option shows the position and tracking of all the players in the 50 kilometer square. This map displays the direction and viewing triangle of the driven vehicle as well as the position and movement of the remaining vehicles. The statistics and data concerning the driven vehicle are displayed in a window at the top of the screen. Speed, pitch, roll, number of remaining rounds and remaining fuel are a few of the statistics shown. Players control their chosen vehicles through several interface devices including a button/dialbox, keyboard and SpaceBall.

The Spaceball is a single device used to control an object's six degrees of freedom. The pick button on the SpaceBall fires the appropriate weapon associated with the vehicle being driven. Applying pressure to the SpaceBall adds to the thrust in the applied direction; twisting the Spaceball changes the object's orientation. This allows the player to turn, move forward, backward, up and down very quickly. Since the system runs in real-time, reactions to events, not just the detection of them, have to take place very quickly. This includes following terrain contours, reacting to input from the user and responding to changes in the 3D world. As long as no collision occurs, the displays on the original NPSNET, NPSNET-1, are realistic.

Purpose And Goals Of Work

NPSNET-1 does not detect nor respond to vehicle collisions. Without collision detection and response, the realism of NPSNET-1 is poor. Even with texturing, environmental effects and realistic looking vehicles, the virtual world falls apart the first time one vehicle drives through another. Our work, implemented in NPSNET-2, detects and responds to collisions between objects in real-time. Detection is sufficiently fast to allow the time needed to respond properly. Response time is dependent upon the level of physically-based modeling.

Physically-based modeling is the process of giving objects the characteristics they actually possess and making those objects react to the forces that influence them in the real world. Transient events are those inputs that act upon the models to change the characteristics of the models. For example a missile impacting upon a tank would definitely change the tank's representation. Characteristics include things such as: spring forces, moldability, rigidity, weight, gravity, explosive potential and much more. Transient events include collisions, explosions, terrain modifications and anything else that affects the physically-based model of either the world itself or the individual objects within that world. There are physically-based models on the market with realistic texturing, collision detection and response. However, very few are done in real-time.

Achieving real-time collision detection and response is the primary goal of this work. Secondary goals include compatibility with SIMNET, realism in the responses and compatibility with future hardware upgrades. Detection of all collisions is another secondary goal. This goal may not be necessary though since realism can be achieved by only responding to those collisions that occur within the viewing area of the user.

Below covers previous attempts to solve the problem of collision detection and response. Other solutions to this problem are covered including interpenetration of bounding

boxes and spheres, physically-based modeling and simple spring forces.

A description of NPSNET-2's collision/response module is presented. Several steps have been taken to limit the number of objects checked for a collision. The implementation of the module is discussed along with the various algorithms and thought processes that went into its design.

The final sections cover results and conclusions including requirements and suggestions for future work in the area of real-time collision detection and response for NPSNET-2.

Other Related Work

There are several papers that cover non-real time approaches to collision detection and response: [Moor88], [Hopc83] and [Terz87]. There are also a few works that cover real-time approaches: [Garv88], [Hahn88] and [Uchi83].

A system for an interactive battlefield simulation is SIMNET [Garv88]; however, it is prohibitively expensive and only runs on hardware specific for each type of vehicle, whereas NPSNET uses one general purpose simulator for all vehicles. SIMNET has a collision detection and response system which is only a small part of the overall system just as this work is only a small part of the NPSNET system. NPSNET complements SIMNET as a general battlefield simulation system; however, there are other existing simulation and collision detection papers which approach the problem of collision detection and response specifically.

An example is a paper by Moore and Wilhelms [Moor88]. It discusses the issue of collision detection and response specifically and goes into detail about both flexible and solid surfaces. The algorithm presented in this paper tests to see if the points of one object are inside the points of another, and if they are, a collision has occurred. Two algorithms for collision detection are given in Moore and Wilhelms' paper, each of which is broken down into two parts. One part tests for planar penetration, and the other part tests for edge penetration. The results of both algorithms are then passed to a collision response algorithm. The algorithm then determines an appropriate response to the collision. The response concentrates on giving new linear and angular velocities to the objects involved. The authors take two approaches: one for objects at rest with forces acting upon them, such as gravity and mass and a second approach for moving objects. The objects at rest respond with spring-like reactions while the moving objects have to be analyzed to determine the appropriate response. Physically-based modeling is discussed and partially implemented. The paper's final solution to the problem of collision response is to use a dynamic approach which can access either the spring force or the analytical method. The biggest drawback to the paper is that the implementation is not real-time.

Another paper is *Collision Detection in Motion Simulation* by Uchiki, Ohashi and Tokoro [Uchi83]. It uses an independent process, a space occupancy method, which detects when spheres, which enclose objects, occupy the same space. Each object is sent a message whenever it tries to occupy a space that is already occupied by another object's sphere. Consequently, message passing is the key to its success. It has an additional feature that makes it unique in that it also passes the point of the collision to the collision detector. This is an important bit of information that is essential to collision response. To properly react to a transient event, the collision point must be known. Again this is a characteristic of physically-based modeling and one that must be preserved in order to accurately and realistically display interaction among objects in the physically-based world.

The paper by Hopcroft, Schwartz and Sharir [Hopc83] provides an algorithm for determining whether a collision has occurred between two objects in three dimensional space. The paper uses spheres to determine intersections between objects. Every object within the paper's model is enclosed within a sphere. The basis of the paper is to determine if any two of those spheres intersect. It also provides a computational complexity analysis for the algorithm along with the mathematics involved in calculating the intersection. Hopcroft's paper also contains the data structures used to create the efficiency of their method. Sorting and placement of the sphere locations and radii are an important part of the method and contribute greatly to the results obtained.

In Hahn's paper [Hahn88] an overview and a limited implementation for a computer animation system to model 3D moving objects is presented. Hahn's paper goes into detail on the physically-based modeling of the objects and the methodology for creating realistic movement of those objects. The paper provides a method for computing the motion of objects by merging not only dynamics but kinematics as well. It allows for interaction between objects that includes collision detection and response. If a collision has occurred, then the collision point along with the backup vector is sent to an analyzer which determines the appropriate response. The response is limited to a bouncing effect at a new velocity and angle. This is the major shortfall of the paper and where the physically-based modeling stops. Responses are built into a table of script files and are limited so that a true response may not be given but whatever comes closest to matching the preprogrammed response.

In *Elastically Deformable Models* by Terzopoulos, et al [Terz87] the problem of deformable objects is discussed. These are objects which would not normally get penetrated but would respond by giving in or bending away from the collision point. Objects of this type include things like paper, rubber and other flexible materials. The paper deals exclusively with deformable objects. *Elastically Deformable Models* does demonstrate what occurs in response to different types of forces, constraints and other objects. The paper promotes the use of dynamic models which react to transient events based upon the principles of applied physics.

A predecessor to NPSNET is the moving platform simulator (MPS) series. It has three versions, with versions 2.0

(MPSII) [Winn89] and 3.0 (MPSIII) [Chee90] each having collision detection. The detection consists of a 2D check for nearness of other vehicles or platforms. If a platform comes within a certain range of another platform then both platforms are killed. There is no check for non-platforms, such as trees, bushes, etc. Additionally, the only response is to kill the vehicles, not damage them or bounce them off of each other.

The fundamentals of detecting collision points are contained in the book *An Introduction to Ray Tracing* [Glas89]. The book covers not only the fundamentals but the specifics of finding intersection points for collisions.

Although, several other programs and systems exist that perform collision detection and response, few do so on SGI IRIS graphics workstation hardware and none do it in real-time. SIMNET comes closest to meeting these objectives but works only on its own particular set of hardware.

Program Implementation

NPSNET-2 runs on any graphics workstation with the GL libraries but has been developed on the IRIS workstations, including the IRIS 4D/120 GTX, 4D/70 GT and the 4D/240 VGX. It is written in Kernighan & Ritchie C [Kern78]. The NPSNET system involves real-time response in a battlefield simulation of land, sea and air forces. The system is networked, via Ethernet, to allow for multiple players to interact. NPSNET performs realistic animation of explosions involving direct and indirect hits by ordnance; collisions of vehicles with other vehicles and terrain features; and terrain modifications such as craters and destroyed trees. NPSNET is relatively inexpensive in comparison to SIMNET. Moreover, NPSNET uses one general purpose simulator to operate on the entire battlefield while SIMNET uses a different type of simulator for each different type of vehicle/platform. This allows any user to sit down at one terminal and become any vehicle in the simulated world. The user can select a different vehicle to operate simply by pressing a button rather than switching hardware. Due to the generic application of the collision detection routines, NPSNET-2 continues to perform in real-time regardless of the simulated vehicle.

World Segmentation

NPSNET's virtual world is divided up into gridsquares of a constant size based upon the actual terrain features of the database from the SIMNET Database Interchange Specification (SDIS) [Lang90]. The gridsquares are small, 125 meters square. Associated with each gridsquare are both fixed and moving objects.

Collision Detection

Collision detection contributes significantly to virtual world realism. Objects passing through other solid objects makes the world unrealistic. A possible solution to this problem would be to prevent interpenetrations by bouncing objects off of each other after any contact, but this is rarely accurate. Another possible solution is to destroy the objects involved in collisions. A third option is to combine these two solutions along with varying stages of damage to involved objects depending upon the physical characteristics of the involved objects. That is the approach taken by our work.

Against Fixed Objects

The algorithm for collisions with fixed objects constantly checks moving vehicles to determine if a collision has occurred. The position of the moving vehicle is updated constantly. Consequently, as soon as a vehicle is moved and its position is updated, it is checked for a collision. In order to maintain a real-time speed, the scope of the collision detection is severely limited. A collision with fixed objects is checked only if the moving vehicle is below a threshold elevation. All fixed objects are in some way attached to the terrain and thus below that threshold elevation. If an object is below that elevation, NPSNET-2 runs through a linked list of fixed objects which are attached to the current gridsquare. This is a quick check since there are relatively few fixed objects in any one gridsquare (Table 1).

Against Moving Objects

A collision with other moving objects is more complicated since any other moving vehicle or object has the potential for colliding with the vehicle we are checking. The potential exists for checking up to 500 vehicles and any of their expendable weapons. Consequently, the scope of the collision detection range has been limited in several ways.

As soon as each vehicle is moved, its position is checked against the position of the neighboring vehicles. If the X or Z position of any other vehicle is within 100 meters of the checked vehicle then those two vehicles are sent to the second level check. At the second level check, the distance between the two vehicles is calculated. If this distance is less than the combined radii of the two vehicles, then a collision has occurred and the third level collision check is done. A rudimentary form of ray tracing determines the actual point of collision.

If worst case numbers are used to determine the implicit range limitations of all vehicles, it can be shown why this culling is fairly accurate. Reasonable speed limitations of the various types of vehicles are used to calculate worst cases for each (Table 2). For example, a ground vehicle with a forward velocity of 60 kilometers per hour travels 1000 meters per minute or 16.6 meters per sec. At a frame rate of 10 frames per sec, this is equivalent to 1.66 meters per frame. Since the vehicle positions are updated each time before the frame is displayed, they are also checked for a collision. A ground vehicle would have to travel at approximately 1000 KPH to completely traverse two gridsquares in one second. Consequently, the movement across more than two gridsquares

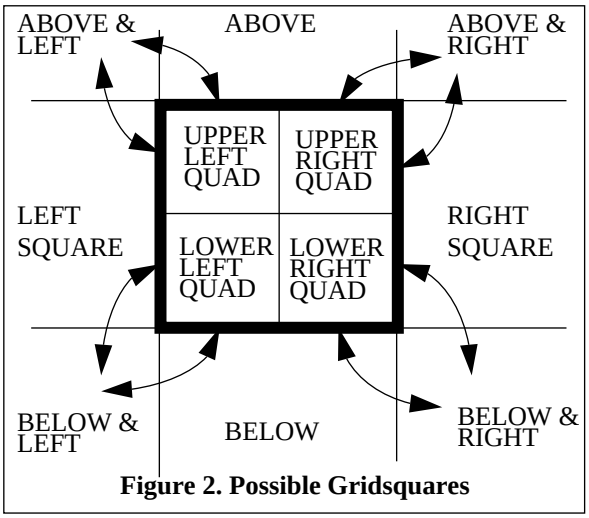

Figure 2. Possible Gridsquares

within one tenth of a second, one frame, is impossible (Figure 1).

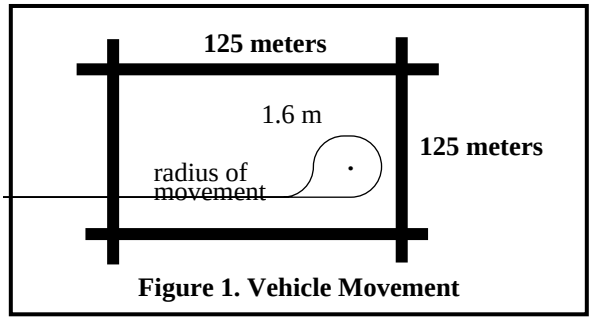

Figure 1. Vehicle Movement

The distance for the first level check is used as a rough approximation for proximity of other vehicles. The gridsquares that can be reached by the vehicle within one frame are checked (Figure 2). The limitation of 100 meters ensures an efficient culling for collision detection and allows the time needed for collision response.

Collision detection is accomplished by determining if one object has interpenetrated another. The most obvious way to determine if a collision has occurred is to create a bounding box or sphere around each object and if that surface is penetrated, then a reaction must occur. Both methods are simple to implement, but the sphere implementation is slightly faster.

The radius used in the spherical check is the maximum distance from the center of the object to the furthest outer vertex. The bounding box uses a maximum and minimum value, not necessarily the value at that part of the object being penetrated. In the collision response portion of the system, the actual object's penetration point is determined. A slightly smaller value than the actual radius of the object is used for the radius. This produces a more realistic collision possibility since it increases the likelihood of an actual collision of the checked objects and not just their spheres. Once the collision has been detected, the extent of damage and collision response are determined.

Collision Response

Collision response is handled by a function which takes in the two involved objects as arguments and determines the impact of the damage upon the them. Many variables must be taken into account to include speed and angle of impact, mass of the objects involved, explosive potential, resistance to destruction, moldability of the objects, rigidity and fabricated spring forces which determine the bouncing-off effect and likelihood of survivability. Each of these factors is weighted in order to provide as realistic an effect as possible while maintaining the environment in real-time. For example, if a tank runs directly over a tree quickly, there should only be a stump remaining if the vehicle operator were to turn around and look backwards. Additionally, if two tanks were to collide at 20 miles per hour, there would probably be a large dent in both along with a severe bouncing effect if the angle of impact was small. If the angle of impact was severe, then both tanks would sustain a large amount of damage. In a real situation, there would be several visual effects that would occur simultaneously in response to the impact. Special effects such as smoke and fire are included.

Fixed Objects

The implementation of collision response requires the input of the two objects involved. A basic assumption that was made was that collisions between more than two objects do not occur very often. Associated with each object, both fixed and moving, are radii that determine the sphere size for the collision checks. For a moving vehicle colliding with a fixed object, there are only a few basic cases:

- ❏ The vehicle is undamaged and destroys the fixed object.
- ❏ Both the vehicle and fixed object are damaged.
- ❏ Neither the vehicle and fixed object are damaged.

In all cases, the fixed object does not move; however, its appearance may change. If the fixed object is large and heavy, such as a building, then the moving vehicle is probably damaged. If the fixed object is small and light, like a small tree or stop sign, then it will be destroyed. A more complex issue arises when two moving objects impact with each other.

Moving Objects

In the case where two moving objects impact, all of the physically-based modeling characteristics of each object must be considered. The collision point must be known to create realistic responses in the involved objects. The collision point determines the point for any type of bending, crumpling and molding. Moreover, if the point of collision is part of a wall that is interconnected to several other walls then there will have to be corresponding responses in those interconnected walls. The only way to find the collision point is through ray tracing.

The first ray is shot from the center of a moving object towards the center of an adjacent object to determine a possible point of collision. This collision may simply be between the bounding spheres of the two objects and not the actual objects themselves. The intersection between the first ray and the second object's bounding sphere is used to specify the direction of a second ray originating from the adjacent object's center.

The second ray determines if one of the object's actual polygons was penetrated. This second ray is the ray used in Haines' algorithm. This algorithm from Glassner [Glas89] was adapted for use in the collision point determination. It involves running through the list of polygons that comprise the adjacent object and determining if the second ray intersects the plane containing the polygon. If no intersection is found once all of the polygons have been checked, then only the spheres were penetrated and not the objects themselves (Figure 3).

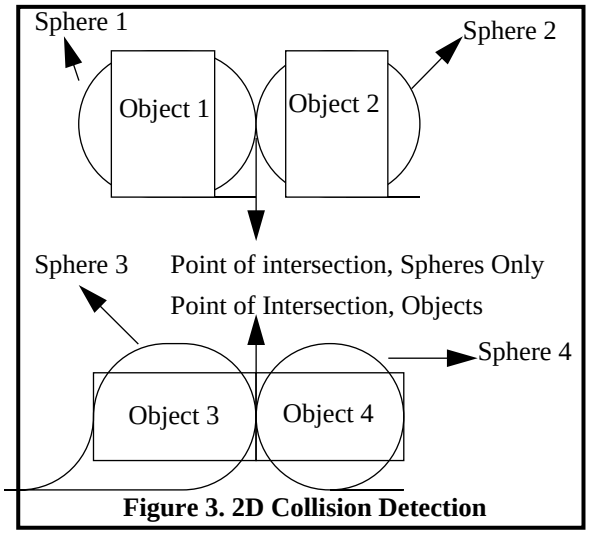

Figure 3. 2D Collision Detection

If the plane that the polygon lies in is intersected then the polygon itself must be checked for an intersection. This is where Haines' adaptation of the Jordan Curve Theorem is implemented. The Jordan Curve Theorem simply states that if a point lies inside a polygon and a line is drawn from that point to another coplanar point outside of the polygon then it will intersect the polygon edges an odd number of times. Conversely, if the polygon edges are intersected an even number of times, then that point lies outside the polygon (Figure 4). The polygon vertices, along with the line segment from the possible polygon intersection to the adjacent object's bounding sphere intersection, are projected onto a common plane. The projected line segment is checked against the edges of the projected polygon for number of crossings. The algorithm is efficient since all edges are either rejected with no intersection at all or accepted as intersected. The algorithm also avoids the problem of points that lie exactly on the edge by placing those points either inside the checked polygon or outside it. The key to making the algorithm work is to determine the dominant coordinate, and to then work only in that coordinate's plane. This simplifies the process enormously and allows for a much faster implementation in only two dimensions.

Reactions

The proper response is performed by comparing the characteristics of two objects involved in the collision. The first check is to determine whether or not the objects involved are fixed or mobile. A few general guidelines are applied to all collisions. The larger massed object inflicts more damage on the smaller massed object. The fixed object has no ability to shift away from the point of contact and consequently suffers more damage than a mobile object with the ability to spring away (Figures 5a

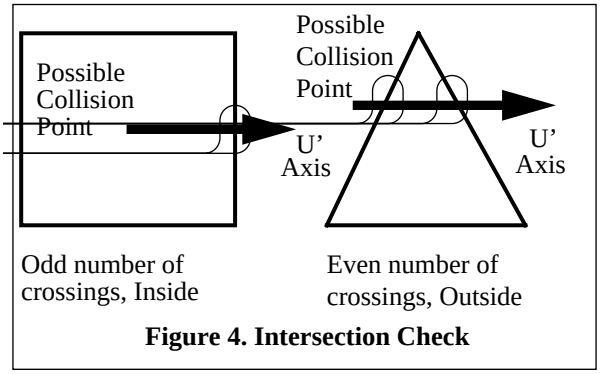

Figure 4. Intersection Check

nad 5b).

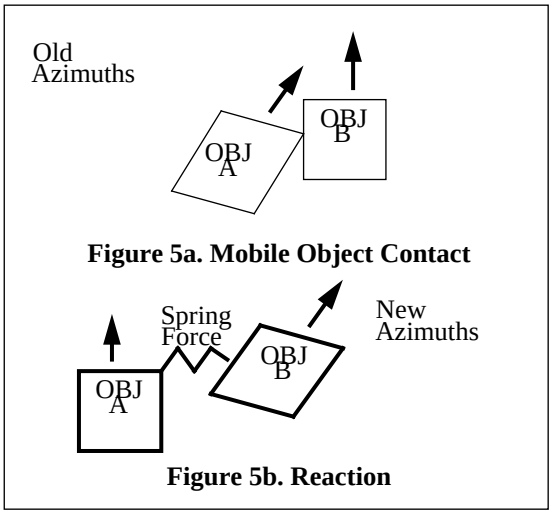

Figure 5a. Mobile Object Contact

Figure 5b. Reaction

A large fixed object, such as a bunker or large rock, can withstand a much larger force of impact than a small fixed object. A large fixed object also inflicts much more damage to the mobile object that struck it. A small mobile object suffers damage if it is hit by a large mobile object at angles that are near multiples of 90 degrees. At smaller angles, even small vehicles are able to bounce away from the impact with a minimum of damage. Consequently, the collision response is limited to a few instances. For fixed objects, the responses include several degrees of damage, based upon the speed and mass of the colliding object. Up to three levels of damage plus the original undamaged fixed object are available for display after a collision. For mobile objects, the response depends upon the angle of impact as well as the speed and mass of the two involved objects. The mobile object reacts by either bouncing away or being destroyed and exploding. In the special case of contact by munitions, the only response is an explosion.The limited number of options available for the response to the collision keep the response fast to maintain the real-time criteria. The collision point and direction of travel are passed to another module that handles physically-based modeling of object movement. This function's implementation can be seen in [Mona91]. That work uses all of the physical characteristics of the object to create the more accurate response that goes beyond the scope of this work.

Performance

The performance of NPSNET-2 is not affected by the addition of the collision detection and response modules. The response time for detection of fixed objects is adequate regardless of the speed of the moving objects. However, for collisions between two high speed objects, collision detection is sometimes slow. When vehicles are traveling at high speed, i.e., faster than about 216 KPH each, they pass through each other rather than colliding. For example: two tanks, each with a radius of three meters, would have to travel six meters in one frame to do this. This is a totally unrealistic speed for a land or sea vehicle but not for an air vehicle. This is due to the inability of the functions to calculate the positions of both vehicles quickly enough to realize that a collision was supposed to have occurred. A time interpolation of the movement of the vehicles will solve this.

Achievement Of Goals

All collisions between fixed and moving objects in NPSNET-2 are detected and respond in real-time and in a realistic manner. The collisions between moving objects is adequate due to the normally slow speed of tactical land vehicles. The effects are realistic and are close to what would occur in the real world.

Conclusions

Realism for simulation is maintained by allowing correct physically-based modeling characteristics to occur in response to transient events within the virtual world. The real-time collision detection and response allow the user to interact with the virtual world and with other networked players.

One of the system's limitations is that it cannot detect collisions between two quickly moving objects. Time interpolation needs to be integrated into the system in order to detect all collisions between two moving objects. Moreover, the system will only detect collisions between two objects, whether they are moving or not. An implied limitation is that all objects in the world are spherical, when in fact, few are. Detections are done on spheres and therefore have a margin for error on the virtual objects. The final limitation is that fixed objects that are large, in comparison to the size of the gridsquare, and are located near the borders of gridsquares may not be detected until after they have been penetrated (Figure 6). However, there are only a small number of these objects in the vir-

tual world.

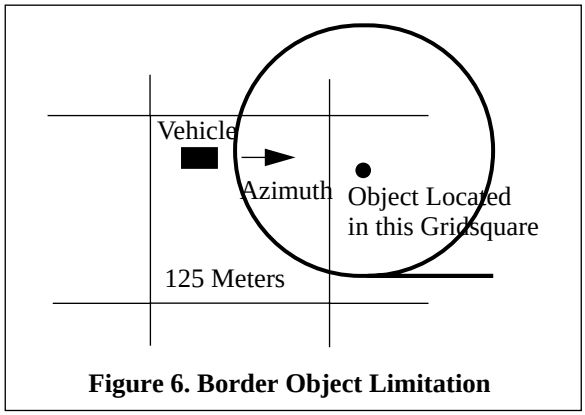

Figure 6. Border Object Limitation

Ideas For Future Projects

Only a few of the physically-based modeling characteristics were used in determining the response to collisions. Obviously, the remainder of those characteristics can be added for more realism. Actual laws of physics were also avoided due to their computational intensity. However, for every area that is added to obtain more realistic affects there is a cost in time. Too many will cause the real-time constraints to be exceeded and cost more than the system can afford. Faster hardware and/or software will allow these constraints to be met. Future work is needed to include all of the physically-based characteristics.

Negative #2

Figure A.1-A Typical Scene From Inside a Vehicle in the Virtual Hunter-Liggett

Negative #3

Figure A.2-Imminent Collision with Tree

Negative #6

Figure A.3-Slow Speed Collision with a Tree

Negative #7

Figure A.4-High Speed Collision with a Tree

Negative #9

Figure A.5-Imminent Collision with Tower

Negative #12

Figure A.6-Slow Speed Collision with Tower

Negative #10

Figure A.7-High Speed Collision with Tower

References

[Chee90] Cheeseman, Curtis P., *Moving Platform Simulator III: An Enhanced High-Performance Real-Time Graphics Simulator With Multiple Resolution Display and Lighting*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1990.

[Garv88] Garvey, Richard E., Jr., and Monday, Paul, "SIMNET (SIMulator NETworking)", BBN Systems and Technologies, Bellevue, WA, July 28, 1988.

[Glas89] Glassner, Andrew S., editor, *An Introduction to Ray Tracing*, Academic Press, San Diego, CA, 1989.

[Glas90] Glassner, Andrew S., editor, *Graphics Gems*, Academic Press, San Diego, CA, 1990.

[Hahn88] Hahn, James K., "Realistic Animation of Rigid Bodies", *Computer Graphics*, Vol 22, no. 4, pp. 299 - 308, August 1988.

[Hopc83] Hopcroft, J. E., Schwartz, J. T. and Sharir, M., "Efficient Detection of Intersections Among Spheres", *The International Journal of Robotics*, Vol 2, no. 4, pp. 77 - 80, Winter 1983.

[Kern78] Kernighan, Brian W., and Ritchie, Dennis M., *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[Lang90] Lang, Eric and Wever, Pete, *SDIS Version 3.0 User's Guide: Interchange Specification, Class Definitions, Application Programmer's Interface*, BBN Systems and Technologies, Bellevue, WA, August 1990.

[Mona91] Monahan, James G., *NPSNET: Physically-Based Modeling Enhancements to an Object File Format*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1991.

[Moor88] Moore, Mathew and Wilhelms, Jane, "Collision Detection and Response for Computer Animation", *Computer Graphics*, Vol 22, no. 4, pp. 289 - 298, August 1988.

[Terz87] Terzopoulos, Demetri, Platt, John, Barr, Alan and Fleisher, Kurt, "Elastically Deformable Models", *Computer Graphics*, Vol 21, no. 4, pp. 269 - 278, July 1987.

[Uchi83] Uchiki, Tetsuya, Ohashi, Toshiaki and Tokoro, Mario, "Collision Detection in Motion Simulation", *Computers and Graphics*, Vol 7, no. 3-4, pp. 285 -293, 1983.

[Winn89] Winn, Michael C., and Strong, Randolph P., *Moving Platform Simulator II: A Networked Real- Time Simulator with Intervisibility Displays*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, June 1989.

[Zyda91] Zyda, Michael J., and Pratt, David R., "NPSNET: A 3D Visual Simulator for Virtual World Exploration and Experimentation",*1991 SID International Symposium, Digest of Technical Papers*, Society for Information Display, Playa Del Rey, CA, pp. 361 - 363, May 1991.