2001

Increasing efficiency in the simulation of a dynamic system of objects in 3-D space

Paulo, Eugene P.; Malone, Linda C.

Simulation Practice and Theory, Volume 8, 2001, pp. 473-490
https://hdl.handle.net/10945/43386

ELSEVIER

Simulation Practice and Theory 8 (2001) 473–490

SIMULATION PRACTICE AND THEORY

www.elsevier.nl/locate/simpra

Increasing efficiency in the simulation of a dynamic system of objects in 3-D space

Eugene P. Paulo [a,*], Linda C. Malone [b]

[a] *Department of Operations Research, Naval Postgraduate School, Monterey, CA 93940, USA*
[b] *Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, FL 32816 USA*

Received 10 March 2000; received in revised form 1 December 2000

Abstract

This article examines sectoring as a methodology of improving the efficiency of discrete event simulation of autonomous object movement in 3-D space. Efficiency is defined as the ability to reduce the amount of pairwise comparisons needed to completely execute the intended system. Sectoring involves partitioning the simulation trajectory space and allows for a reduction of the number of queries required between objects in order to determine upcoming events. However, the crossing of sector boundaries becomes an additional event to track and compute. A simulation of moving objects in 3-D space was constructed using MODSIM™. This simulation was verified and used to gather significant output data. The results indicated that, within the specific experimental range, sectoring provides up to 52% improvement in simulation efficiency when compared to no sectoring. Published by Elsevier Science B.V.

Keywords: Sectoring; Discrete event simulation; Simulation efficiency; Autonomous object movement

1. Introduction

The movement of autonomous objects in 3-D space is representative of many significant physical systems. Examples of these systems are the movement of weapon systems in military combat, air traffic patterns around any airport, and in the medical field, where movement of fluids, such as blood or serum, through the body must be simulated.

* Corresponding author.
E-mail address: eppaulo@nps.navy.mil (E.P. Paulo).

0928-4869/01/$ - see front matter Published by Elsevier Science B.V.
PII: S0928-4869(01)00029-5

This article focuses on a research effort that examined a method of improving the efficiency of simulated autonomous object movement in 3-D space. While improved computer hardware has resulted in much faster computing capabilities, improvements in the algorithms that drive the discrete event simulation could lead to an increase in efficiency and, therefore, reduced computer run-time.

Efficiency is defined as the ability to reduce the amount of pairwise comparisons needed to completely execute the intended system. This definition has been used by other researchers examining this type of problem [6–10]. In this research, efficiency is measured and examined in two ways. The first is through the development and analysis of a mathematical model of the system, which is described in a complementary article [5]. The second method is the use of simulation, which is the focus of this article.

Recent research into simulation efficiency as defined above has found merit in the concept of sectoring in a 2-D environment [7–10]. For spatial objects to move autonomously and remain accounted for, the location vector of all other objects within the boundaries of the trajectory space must be tracked. This has routinely meant that scheduled events are the result of each object querying every other object. This basic technique is computationally inefficient when the number of objects gets large. The idea of sectoring allows the trajectory space to be split into sectors of certain geometric shapes that require an object to query only other objects that are in its sector or adjacent sectors. While sectoring reduces the number of queries between objects, the crossing of sector boundaries becomes an additional event to track and compute. Therefore, in order for sectoring to be beneficial, the reduction in overall queries between objects (pairwise comparisons) must overcome the additional computational load of tracking the objects as they cross sector boundaries.

Research into sectoring has resulted in several important discoveries. First, the plausibility of two particular sectoring methods, known as fixed sectoring and dynamic sectoring, has been demonstrated in the 2-D billiard balls problem. Fixed sectoring provides a fixed number of identical-sized sectors throughout the entire simulation area for the duration of the simulation. Dynamic sectoring allows the size of the sectors to change depending on the number of objects currently within a sector, attempting to efficiently account for the contrast in number of objects to query versus the necessity of accounting for the crossing of sector boundaries. Dynamic sectoring is thought to be most effective when the objects are not randomly or uniformly distributed throughout the simulation area [10].

The discrete event simulation utilized in this research is an object-oriented representation of identically sized, simple 3-D objects (essentially balls) moving at constant speeds throughout a 3-D simulation space. Different combinations of number of balls and ball radii are examined, with the intent of discovering an experimental region where the use of fixed sectoring provides improvement in computational efficiency over the use of no sectoring. Additionally, throughout the development of this simulation, portability was an important issue. An important objective is that, this simulation code be constructed such that it is portable to various platforms and easily compiled and run by any interested user.

2. Related research in 2-D space

Two studies regarding 2-D sectoring [8,9] had significant impact on this research. The approach used by Harless was to build upon the concept of fixed sectoring in the 2-D billiard ball problem first simulated by Goldberg. The focus of Harless was to consider the issue of non-uniformly distributed autonomous objects and interject the concept of dynamic sectoring. With dynamic sectoring, sectors were subdivided whenever a sector contained more than a pre-specified number of spatial objects. Additionally, sectors were rejoined whenever the number of spatial objects was less than some pre-specified number of spatial objects. Thresholds for determining when to subdivide were determined with some preliminary testing. Using a 2-D billiard ball simulation, Harless demonstrated that the dynamic sectoring methodology can provide a smaller mean execution time for non-uniform distribution of spatial objects than the fixed sectoring methodology [10]. Doescher [8] attempted to validate the opinion of previous sectoring researchers by establishing statistically that fixed sectoring is more efficient than no sectoring in certain circumstances.

3. 3-D simulation development

The simulation developed by Harless [10] provides a modeling platform for this research. Harless developed an object-oriented simulation of the movement of autonomous objects using the three methodologies of no sectoring, fixed sectoring and dynamic sectoring in a 2-D environment. He structured his simulation code in MODSIM II, which is a general purpose, modular, block-structured high-level programming language. It provides direct support for object-oriented programming and discrete event simulation [2].

3.1. Trajectory space and sector representation

The trajectory or simulation, space is defined by Euclidean geometry, with x, y and z directional components representing 3-D space. Therefore, the trajectory space is restricted to only a "block" shape or cube when the x, y and z lengths are equal.

Sector representation is restricted to a certain number of equal-sized blocks. This number of blocks is defined by the number of sector partitions desired along each directional component. If the number of sector partitions is equal for the x, y and z components, then, each sector would be in the shape of a cube.

3.2. Event between ball and boundary

An event between ball and boundary occurs whenever a ball collides with the boundary of the entire simulation region (also called trajectory space) or when the ball crosses a sector. As a result, the discussion of determining possible events between ball and boundary described below represents both possible sector crossings and possible collisions with the boundary of the trajectory space.

Since the sectors are defined as cubes in 3-D simulation, the sector boundaries are obviously rectangles. The representation of these boundaries in the simulation, and how the program determines if a collision between ball and sector boundary occurs, is based on the 2-D simulations of Harless and Goldberg. Therefore, this section is divided into two parts, beginning with a description of the mathematical derivation and representation of a collision between ball and sector in 2-D, followed by how this event is developed for the 3-D simulation.

3.2.1. Ball and boundary collision in 2-D

In his 2-D research, Harless determined sector boundaries by defining the x and y coordinates of the sector corners (as a function of the number of partitions along each directional component) and the vector that represented the distance between the two points. These vectors were created for each sector boundary such that they pointed clockwise in succession around the sector. The sector boundary was therefore a line segment represented in vector notation by the two endpoints (e_1, e_2) and the vector between them $(l = e_2 - e_1)$, as seen in the Fig. 1.

The algorithm used by Harless to represent the collision between a ball and a sector boundary was based on the work by Goldberg. The problem was to first determine whether a ball with a given velocity would hit a sector boundary. If this was true, the time of this event would be calculated.

Goldberg described the ball as having a radius r and known trajectory. The ball hit the boundary when its centerpoint was at a distance r from the boundary. The set of points that make up the boundary was given by

$$\alpha e_1 + (1 - \alpha)e_2, \tag{1}$$

when $0 \leqslant \alpha \leqslant 1$. Since l points clockwise around the sector boundaries, the unit vector pointing into the sector perpendicular to l, which was labeled u, was given by

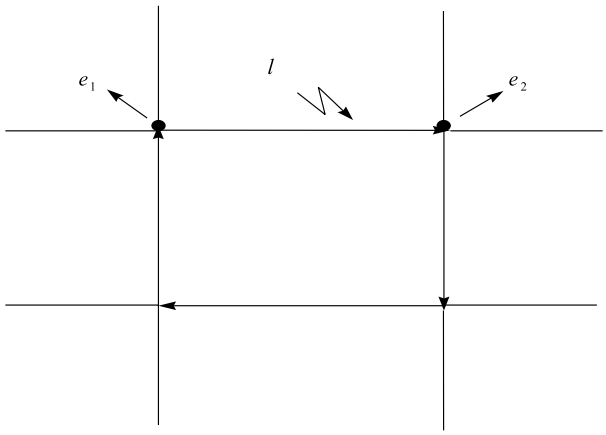

Fig. 1. Sector boundary notation.

$$u_x = \frac{l_y}{|l|}, \tag{2}$$

$$u_y = \frac{-l_x}{|l|}. \tag{3}$$

Let R be a vector in the direction u whose length equaled the radius of the ball, and $R = ru$. Then, the set of points in the sector of concern that were at a distance r from the sector boundary was

$$\alpha e_1 + (1 - \alpha)e_2 + R. \tag{4}$$

Therefore, to determine if and when a ball touches and starts to cross a sector boundary, set the equation describing the movement of the ball equal to the Eq. 4 that described all points at distance r from the sector boundary. The movement of each ball was determined by an initial position, which was represented by $s(t_0)$, and its movement over time $(t - t_0)v$, with v representing the ball's velocity. The equation became

$$s(t_0) + (t - t_0)v = \alpha e_1 + (1 - \alpha)e_2 + R \tag{5}$$

and after breaking this down into simultaneous equations of the x and y components and solving for t, if $0 \leqslant \alpha \leqslant 1$, an event involving a boundary and a ball occurred.

3.2.2. Ball and sector interaction in 3-D

For 2-D simulation, these mathematical relationships were relatively straightforward, as demonstrated in an abbreviated manner above. However, determining whether a 3-D ball will collide with a rectangle defined by four corners and four vectors serving as edges of the sector boundary, and calculating the time of collision, is more complex.

Additionally, after solving this problem mathematically, it is necessary to develop a code that would run in a version of Harless' 2-D simulation that was modified to portray a 3-D environment. Fortunately, a solution is found that is mathematically correct in calculating the possible occurrence and time of a ball touching a boundary, without adding great change to the simulation code.

A straightforward, yet accurate, method is to consider the projections or shadows, of the 3-D ball onto 2-D planes. Two brief points, regarding the Euclidean geometry inherent in the dimensions and boundaries of the sectors and the overall trajectory space, require further emphasis. First, in 2-D, the sector has four boundaries, with two boundaries representing the upper and lower limits of the x component of ball location within the sector, while the other two represent the upper and lower limits of the y component within the sector. However, in 3-D, the sector becomes cubic with six boundaries, with the additional two representing the upper and lower limits of the z component.

The 3-D algorithm checks the six sides in progression for the occurrence of a collision and the time of collision. Using the same mathematical steps described above for 2-D, the ball is first projected onto the xy plane, and the first four sides are

checked. This projection means that only the x and y coordinates of its location and velocity are considered. The sector boundary is also represented by only its x and y coordinates, making it a line segment such as described in the section above. If the ball does not touch a boundary on the xy plane, then the ball will not touch the respective boundary in the 3-D environment.

For the final two boundaries, the ball is projected onto the yz plane, and then four sides are each checked for possibility of a hit by a ball, and the corresponding time of the hit. Since the final two boundaries are the minimum and maximum of the z component of ball location within the sector, inclusion of the z component of the velocity vector and ball location allows for accurate calculation. After all six boundaries are checked, their respective times of collision are compared, with the shortest becoming the actual interaction time of the ball and boundary in 3-D.

3.3. Collisions between two balls

As with the interaction between ball and boundary, the algorithms and subsequent equations that describe the collision between two balls in the 3-D simulation is also based on work by Goldberg in his 2-D simulation. Therefore, Goldberg's development of these equations in 2-D is seen in the next section, followed by a section discussing how they are extended to apply in the 3-D simulation.

3.3.1. 2-D collisions between balls

Given the trajectories of two balls, the problem becomes determining if and when the two balls will collide. The future location of each ball is determined by an initial position in the calculation, which is represented by the function $s(t_0)$, and its movement over time $(t - t_o)v$. That position can therefore be written

$$s(t) = s(t_0) + (t - t_0)v. \tag{6}$$

Goldberg makes the following substitution in order to simplify the problem.

$$D = s(t_0) - t_0 v. \tag{7}$$

This allows for a simpler expression for the trajectory, which is

$$s(t) = D + tv. \tag{8}$$

Adding subscripts to label the trajectory or future position, of two balls results in the following equations for each ball:

$$s_1(t) = D_1 + t_{1,0}v_1, \tag{9}$$

$$s_2(t) = +D_2 + t_{2,0}v_2. \tag{10}$$

The distance separating the balls ($d(t)$) satisfies the equation

$$[d(t)]^2 = |s_1(t) - s_2(t)|^2. \tag{11}$$

Let the radius of the balls be r_1 and r_2. The condition for collision is that the balls touch or that

$$d(t) = r_1 + r_2. \tag{12}$$

Substituting for $d(t)$ gives the following equation.

$$(r_1 + r_2)^2 = |s_1(t) - s_2(t)|^2. \tag{13}$$

Goldberg expands the right-hand side into expressions for vector components, and after some algebraic manipulation and solving for a quadratic expression, the result is the following equation for t.

$$t = \frac{-\Delta v \Delta d - \sqrt{\alpha}}{|\Delta v|^2}. \tag{14}$$

In this equation, $\Delta v = v_1 - v_2$, $\Delta d = D_1 - D_2$, and $\sqrt{\alpha}$ is a simplification of the term that forms the square root portion of the basic formula to solve quadratic equations, which is

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{15}$$

If $\alpha \leqslant 0$, then the balls do not touch. Otherwise, the balls collide and the time of collision is t.

3.3.2. 3-D collision between two balls

Again, the way to solve this problem in 3-D is to consider the projections of the balls onto the 2-D planes. The difference here is that, for each possible collision between balls, their projection onto the xy, xz, and yz planes are checked, in order, for possible collisions using the same mathematical equations described in the previous section. If it is determined that on each of the three planes there will be a collision, the latest time of the three becomes the recorded collision time in the 3-D simulation.

In a later section, the model verification effort shows that there is a flaw in the 3-D algorithm described above. Nevertheless, this error appears to be negligible in its effect on this study and its purpose.

Of the three possible events described above, two require some mathematical method to determine new velocity vectors for the balls involved. Specifically, collisions between two balls or between a ball and the boundary of the trajectory space result in new movement directions. It is important to note that while both sets of 3-D simulation code concerning the results of collisions have Goldberg's 2-D equations as a basis, the difficulty of correctly incorporating them fully into 3-D equivalent equations lead to certain compromises. The specifics of these compromises and the subsequent method for determining the new vectors for each of the two collision events are described below. Additionally, conservation of energy and conservation of momentum are both assumed during the collisions.

3.4. Resolving collision between ball and boundary

In 2-D, resolving the collision between a ball and a boundary is somewhat complicated if the boundaries are of irregular shapes or not parallel to the x- or y-axis.

However, if the boundaries are parallel to the x- and y-axes, the calculation becomes simply a change of sign of one component of the velocity vector. In this specialized case, whichever directional component is limited by a boundary (such as the highest allowed value for y), that component of the velocity vector changes signs.

For 3-D, the same principle is used. For the purposes of this research, all sector shapes are cubic, with the six boundaries of the cube representing the upper and lower limits of each of the three directional axes. Therefore, a collision in 3-D between ball and boundary is resolved by leaving two of the vector components the same, while changing the sign of the component that is represented by the respective boundary.

3.5. Resolving collision between balls

Many classical physics and mechanics texts describe a collision between balls in two dimensions. Both the cases of elastic collisions and inelastic collisions are often discussed in detail. Fig. 2 illustrates this phenomenon.

Fig. 2 shows that the force of impact occurs where the balls touch, which is represented by the two vectors $v_{1\text{norm}}$ and $v_{2\text{norm}}$. It also shows that the velocities of the two balls following the collision are a function of these vectors by displaying their geometric relationships. While resolving this process in 2-D is a straightforward mathematical process, it becomes more complicated in 3-D due to geometric considerations.

To simplify the process, it is assumed when resolving the collision that the balls are points without a radius. The conservation of energy and momentum are still assumed to be true during each collision. The outcome of this simple calculation is that a collision of two balls in this 3-D simulation results in an exchange of velocity vectors. It is important to note that while ball radius is not part of the calculation for resolving the collision between balls, radius is still very much a part of the determination of whether a collision would occur and the time of the collision.

3.6. Model verification

Model verification refers to the ability of a simulation model to perform in a way that is expected and sensible [1]. This is accomplished through a debugging of the

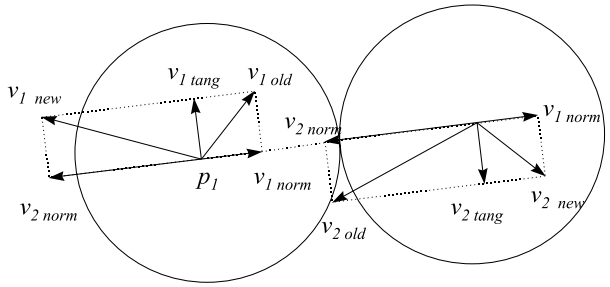

Fig. 2. Resolving ball collision in 2-D using radius.

Table 1
Model verification summary

Behaviors investigated	Simulation output observed	Verified?
Ball collisions conserve velocity	New velocity vectors following collisions have same velocity magnitude as prior to collision	Yes
Ball switching sectors correctly	Switching to correct, adjacent sectors occurs only when ball location is actually on sector boundaries	Yes
Ball collisions with boundary of trajectory space occurring correctly	Ball location during this event at edge of trajectory space	Yes
Ball collisions occur correctly	Magnitude of distance between balls during collisions equal to two times radius	No, has slight errors

simulation and studying its output. During simulation runs, tags were placed in the code in order to produce specific information concerning simulation entities (balls and their sectors) during collision. This information included the x, y and z coordinates of balls, their current sectors and the distance between balls (or ball and boundary) during collisions. Table 1 shows a summary of the simulation behaviors investigated, how they are observed and studied and whether complete verification occurs. Of the four primary behaviors investigated, the check of whether the balls were the proper distance apart during collisions shows some discrepancies and requires further explanation.

The verification effort concerning the correct occurrence of ball collisions reveals a flaw in the algorithm that identifies the next event for a ball. An earlier section described the projection method used to determine whether a collision between balls will occur. The results of the verification process show that collisions between balls in the simulation are observed at ranges of up to 12% greater than two times the radius of the balls. Specifically, at a setting of 1500 balls and radius of 20, 12 ball-to-ball collision events chosen at random occur at distances ranging from 40.01 to 45.32, as opposed to a distance of 40.

Further investigation indicates that the reason for the discrepancy is that the use of the projections has been oversimplified. Consideration is not made for the angles formed by the line segment connecting the center of the two balls at the moment of collision, and each of the three planes. The best case is when one of the angles approaches zero (corresponds to the distance of 40.01). The worst case is when the angles formed by the balls and the three planes are all approximately 45°, which results in the maximum error of 29.3%. This is demonstrated through the use of an equilateral triangle, with the hypotenuse representing the actual distance ($\sqrt{2}$) and the side of the triangle representing the distance measured using the 3-D simulation algorithm (1). The maximum error is found by subtracting the distance measured using the 3-D simulation algorithm from the actual distance and dividing by the actual distance.

The impact of this error in the collision algorithm on the results and conclusions of this study are minimal for two reasons. First, while the exact time of the collisions

will often be incorrect, the collisions will almost certainly occur nevertheless at a slightly earlier or later time in the simulation. Second, since the focus of the study is to compare the benefits of sectoring versus no sectoring in otherwise similar scenarios and conditions, the relatively minor effects of the error are felt equally in both conditions of sectoring and no sectoring.

4. Evaluation of 3-D sectoring simulation

4.1. Experimental design overview

The primary goal in the experimental design is to demonstrate empirically that there are situations where fixed sectoring using cubic sectors provides more computer efficiency than no sectoring in 3-D and to develop a regression model relating the factors affecting efficiency. An additional goal is to try to develop a heuristic that can identify specific sector sizes that provide the most benefit to particular object densities.

Following the development of an acceptable 3-D simulation, an experiment is conducted to evaluate the effects of fixed sectoring on the efficiency of the 3-D simulation of moving autonomous objects. The effects of fixed sectoring are examined over various numbers of moving balls and radii, while using different numbers of sectors. The upper and lower limits of the factors are initially influenced by simulation results from previous 2-D research, with some consideration given to results of the analytical modeling completed in this research. However, the final determination of these upper and lower limits depend on a form of region searching, based on preliminary analysis in this 3-D environment.

4.2. Factors used in the experiment

The main effect factors used in this experiment are number of balls, radius of balls and number of sectors. An additional factor used in this experiment is *density*, which is essentially a third-order interaction term. In a 3-D environment, density is the ratio between the total volume (area in 2-D environment) of the balls in the trajectory space, divided by the total volume of the trajectory space (L^3) itself. L was set at 1200 throughout this experiment. The equation for density is written as:

$$\text{Density} = \frac{N((4/3)\pi r^3)}{L^3}. \tag{16}$$

The influence of density in the 2-D environment [8] led to its inclusion as a factor in this 3-D research. While this led to some collinearity between variables, the experimental analysis provides evidence of the importance of density in the model.

Velocity is not included as a factor in determining the expected number of collisions between two balls, or between ball and boundary. This is due to the inherent dependency among units of velocity, units of simulation time, and units of distance traveled by the balls (velocity = distance ÷ time). Assuming that all balls traveled

with the same velocity magnitude, which is the case in all simulation runs throughout this research, increasing velocity would have the same effect on the number of collision events as increasing the duration of the simulation time.

4.3. Determining the experimental region

Generally, the experimental design concentrates on the response variable, by determining which factor levels provide optimum response. For example, if a manufacturing company is examining a part's molding process in order to reduce shrinkage, the factors involved could be the injection velocity of the heated liquid, mold temperature, mold pressure and back pressure. The intent of the experimental design in this manufacturing example would be to discover which combinations of settings provide minimum shrinkage (which optimizes the response variable), and to change the process settings to these new settings.

However, in this research, the focus moves away from the response variable to one of the factors. Systems that will or can be represented by this simulation model will have a particular fixed or predetermined number of autonomous moving objects with specific radii. As stated previously, 3-D simulations can be used to model military weapon systems on a battlefield, such as tanks or fighter aircraft. In this, and any other physical application, the size of the tank (radius) will be fixed. The number of tanks (balls) will also be fixed for each scenario. Thus, density will be fixed. The only factor value that could be controlled by the experimenter is the number of sectors. Therefore, the intent of the experiment is not to set all the factors at levels such that computer run time is minimized, but rather to input the number of balls, radius, and density to be analyzed, and establish the number of sectors needed to achieve an optimal response variable. As a result, sequential searching in this case is done to find the relationship among the factors and the response variable, and ultimately to find an experimental region where it is possible to tell a study sponsor or proponent the best sector setting depending on the experimental conditions.

An additional consideration in determining the experimental region is to determine what, if any, practical restrictions on the upper bounds of the factor levels will be dictated by computer limitations. Previous research in 2-D showed that computer hardware capabilities affected the number of balls that could be inserted into the simulation.

4.4. Previous research results

The starting point for determining the experimental region is the settings used in previous 2-D research. Doescher [8] was able to demonstrate the significance of sectoring at certain combinations of number of balls (150, 300, 450), radius length (1, 5, 10), number of sectors (1, 36, 100, 144), and density (ranging from 0.01 to 0.10). Doescher showed that fixed sectoring provided increased computational efficiency over a range of density values and that the number of sectors providing the most benefit changed as density changed.

The upper limit of the number of balls allowed in the experimental region is based on computer hardware capabilities. Experimental simulation runs are conducted with 10,000, 7500 and 5000 balls (at $r = 10$ or $r = 15$) without success. At 10,000 and 7500, the huge number of pairwise comparisons required to determine a ball's next event means that running the simulation for just one time unit takes over 72 h to complete, and makes output data too large for the computer to store and display due to memory limitations. With 5000 balls, the output data becomes too large for the computer to store if over two simulation time units are used. It is, therefore, not possible to gather sufficient data at 5000 balls or more, so 2500 balls become the upper limit.

4.5. Sequential analysis

Initial simulation runs use 500 balls and radius of 5. Results demonstrated no improvement over no sectoring in computational efficiency (meaning no decrease in computer run-time) using any number of sectors. It is clear that a sequential search, where various increasing combinations of factor levels were analyzed, would be required in order to provide an acceptable experimental region.

This sequential search means making a series of intuitive sequential steps in order to establish an experimental region. First, the radius of the balls is increased sequentially from 5 up to 20, while the number of balls in the simulation remains at 500. This results in a range of 0.00015 to 0.0097 for the value of density. The trajectory space is cubic in order to maintain some simplicity in the problem, resulting in number of sectors used ranging from 64 (4^3) to 1728 (12^3). However, these trials do not indicate that sectoring provides any computational savings.

The next step is to increase the number of balls in the simulation to 1000, and increase the radius sequentially, using an even greater range, from 5 to 40. This produces density levels from 0.0003 to 0.1551. At higher density levels (based on higher ball radii), sectoring is discovered to provide a reduction in computer run-time. This establishes the lower level for number of balls in the experimental region, and provides a basis for the range of the radius and density factors. The number of balls is increased by increments of 500 up to 2500, again showing a varying amount of increased computational efficiency depending on radius, density and the number of sectors. The number of sectors investigated again ranges from 64 to 1728, but the focus is 216 and 1000, since these two consistently provided the most run time improvement among sectoring options.

4.6. Describing the experimental region and number of trials

As noted earlier, the factors considered in this research are number of balls, radius, density and number of sectors. The process described in the above section determines the necessary ranges in the values of the factors. The response variable is computational time per simulation time unit, which is generated by the simulation following each iteration. Sector crossings are counted and reported. Table 2 provides a summary of the experimental design factor levels and experimental region. However, a full

Table 2
Experimental design factor levels

Factors	Levels
Number of balls	1000, 1500, 2000, 2500
Radius	5, 10, 15, 20, 25, 30, 33, 35, 40
Object density	0.0003, 0.0005, 0.0006, 0.0008, 0.0024, 0.0036, 0.0048, 0.0061, 0.0123, 0.0164, 0.0194, 0.0291, 0.0654, 0.1309, 0.1551, 0.1636, 0.2079, 0.2178, 0.2327, 0.3313
Number of sectors	1, 216, 1000

factorial design is not done due to the fact that the objective is not exactly optimization of the response, as described previously.

4.7. Experimental results

There are 60 simulation runs that are conducted within the experimental range. Output data gathered from each run is the simulation run-time and the number of pairwise comparisons. The number of collisions between two balls, collisions between a ball and the trajectory space boundary, and sector crossings are recorded for most of the runs. Each simulation run (representing a particular setting of number of balls, radius and number of sectors) is initialized by placing the balls in random locations, with random movement directions, within the trajectory space.

4.7.1. Analysis of simulation data

The initial intent of the experimental design, once the factor levels are determined, is to conduct a full factorial design that will generate a data point for every cell. However, this becomes unnecessary and infeasible for some combinations of factor levels. It becomes quite apparent, as simulation runs occur, that certain combinations of balls, radius, and sector numbers within the experimental range will provide no additional information. This is because the most important factor in the 3-D analysis is object density. Regardless of the number of balls and radius of balls that form this "interactive" term, object density proves to be the primary determinant of whether sectoring is beneficial, as well as what number of sectors provides the greatest reduction in simulation run-time.

This conclusion is evidenced by Fig. 3, showing the data output for the experimental region. It demonstrates a clear relationship between sectoring benefits and object density when concerned with the effect of sectoring on computational efficiency. The y-axis of the figure shows the ratio between simulation run-time (assuming one simulation time unit is input) for the case of the "best" sector size versus no sectoring, while the x-axis reflects the object density term for each run. Thus, where y-axis is one, sectoring and no sectoring are identical regarding their influence on simulation run-time. Combinations with ratio greater than one indicate no sectoring is preferable, while those with ratio less than one show sectoring superior to no sectoring.

At object density below 0.01, there is only one instance where sectoring provides improved simulation run-time when comparing to no sectoring. However, as density

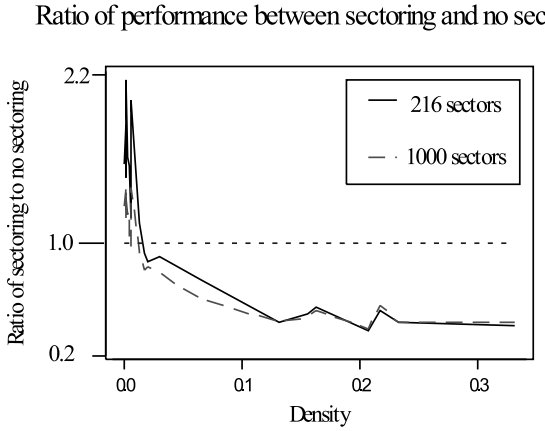

Fig. 3. Ratio of performance between sectoring and no sectoring.

increases past the 0.01 level, sectoring shows increasing benefit. More specifically, the use of 216 sectors provides the largest decrease in simulation run-time compared to any other number of sectors until object density reaches approximately 0.20. At that density and higher, the use of 1000 sectors provides a slight improvement in efficiency over 216 sectors. Using the three density settings in this experiment, 64 or 1728 sectors never results in the most efficient simulation run.

These results lead to the heuristic that is graphically displayed in Fig. 4. Fig. 4 displays a continuum that shows the preferred number of sectors based on object density. The heuristic only applies to the experimental region and to this specific implementation of a simulation that models the movement of balls in 3-D space.

4.7.2. Regression model

A regression model is developed with the intent of further identifying the most important factors, interactive terms, and their relative importance and relationship to run-time. Of particular interest is the object density factor and its inherent interdependency with number of balls and radius.

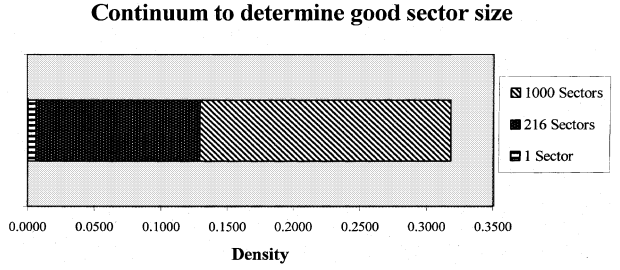

Fig. 4. Optimal sectoring heuristic.

Determination of the best regression model begins with the full model. The full model includes the four factors (*balls radius sectors* and *density*), their square terms, their two-way interaction terms, and all three-way interactions. All input data was coded with appropriate values between −1 and 1, using a method described by Myers and Montgomery [3]. The response variable is the average simulation run-time for one simulation time unit, meaning that if the simulation was run for 100 time units, the overall execution time would be divided by 100.

The selection of the ''best'' regression model is based on achieving the lowest possible mean square error (MSE), while also addressing collinearity. Terms with the highest *p*-value (meaning that particular term did not appear to contribute to the predicted value of simulation run-time) are removed in succession, unless their removal results in a higher value for MSE. Each model term is also checked for a high variance inflation factor (VIF), which is a useful measure of collinearity [4]. Myers and Montgomery [3] suggest that values of VIF lower than 5 allow for a conservative conclusion that collinearity is not a problem in a model. The final model shown in Table 3 results in the second lowest MSE of any regression model calculated from the data, and has no VIF greater than 4.5.

This model shows several important characteristics of the 3-D simulation of moving autonomous objects. First, while the final regression model has the lowest MSE of any model, it includes a term ($balls^2$) with a high *p*-value. This is indicative of the inherent relationship of the factors in the experiment, particular between balls and density, as well as radius and density. Additionally, the factors with the highest positive coefficients in the regression equation are the interaction term $balls * density$ and the first order *balls* term. This indicates that an increase in the number of balls, together with an increasing value of interaction of balls and density, have a large effect on the amount of simulation run-time.

Table 3
Regression results[a]

Predictor	Coef	S.D.	*t*-ratio	*p*	VIF
Constant	70.6	131.5	0.54	0.593	
Balls	1305.6	153.0	8.53	0.000	4.4
Sectors	774.6	198.4	3.91	0.000	4.5
Balls^2	442.3	128.4	3.44	0.001	1.2
Radius^2	179.9	155.3	1.16	0.251	1.1
Sectors^2	1521.0	261.8	5.81	0.000	4.1
Balls∗sectors	−461.7	145.5	−3.17	0.002	2.2
Ball∗density	947.4	162.7	5.82	0.000	3.0
Radius∗sectors	−837.2	109.7	−7.63	0.000	1.1
bsr	−784.3	206.1	−3.81	0.000	2.1
bdr	260.1	183.9	1.41	0.162	2.4
MSE = 445.2	$R^2 = 89.5\%$	$R^2(\text{adj}) = 87.9\%$			

[a] Run-time = 71 + 1306 balls + 775 sectors + 442 balls^2 + 180 radius^2 + 1521 sectors^2 − 462 ball∗sectors + 947 balls∗density − 837 radius∗sectors − 784 bsr + 260 bdr.

4.7.3. Conclusions

The process of searching for, identifying, and closely examining the selected simulation experimental range suggests several important conclusions. First, fixed sectoring does provide improved computational efficiency over no sectoring under certain conditions. In some instances, the improvement is more than 50%, revealing that fixed sectoring has enormous potential.

Second, it is clear that the degree of improvement in efficiency is greatly affected by the corresponding value of density. The heuristic, shown graphically in Fig. 3, demonstrates the effect that density has on the selection of the number of sectors that provide the most benefit. This is also reflected in the regression model, as the interaction term of $balls * density$ has the greatest impact on the regression equation, as evidenced by it having the largest coefficient of an equation with coded variables as factors.

Finally, it seems that there are other factors, or occurrences, involved in the sectoring simulation that remain unexplained. It is not intuitive that only in the experimental range (described above) would sectoring provide increased efficiency, and that the use of 216 sectors would be dominant over the use of a higher or lower number of sectors for much of the experimental range. Obviously, the complexity of the simulation, particularly run time required for various modules and procedures involved in the sectoring process, is not entirely understood and could use further scrutiny.

5. Summary and future research

5.1. Improve the sectoring process

Developing a simulation that portrays different sector shapes, such as concentric spheres, and different types of object distribution, such as a cluster of balls drawn together by a type of attractor, could provide great benefits. The portrayal of a cluster-type distribution would allow analysis of the movement of balls in a different manner, which could represent real-world phenomena such as airplanes approaching and leaving an airport and troop movements near a battlefield. The ability to include concentric spherical shapes in a simulation could make use of the inherent advantage of concentric spheres having much fewer adjacent sectors to check. It would also attempt to discern whether the new constructs involved, such as circular boundaries and spherical coordinates, could be efficiently developed.

5.2. Using density as threshold for dynamic sectoring

In the 3-D simulation we developed, there is no dynamic sectoring capability. We here, refer the 2-D simulation that had previously been developed, and describe the methodology used to allow for dynamic sectoring. Nevertheless, it is seems conceptually straightforward to develop the algorithm and code for 3-D dynamic sectoring, although it may be very time consuming. In addition, the basic premise of dynamic sectoring could be changed, and possibly improved, by establishing an upper and

lower threshold for object density in a sector, rather than number of objects. Some preliminary trials could provide reasonable upper and lower boundaries for density in order to focus simulation runs on the most interesting densities for in-depth research. Results that indicate ideal thresholds may be very valuable, and could result in an increase in computational efficiency under certain conditions.

5.3. Object representation

Another topic for future research could be to improve the representation of spatial objects. Their current portrayal as free moving balls in 3-D space is acceptable at this research level as a basis for the theoretical study of sectoring benefits, but in order to represent a real, physical system, some form of "intelligence" should be assigned to the moving objects. This intelligence could initially be a simple addition to how a ball collision is resolved, possibly resulting in certain directional movements. Also possible is the development of a method of dynamically changing velocities to portray objects moving at different speeds. It would take some effort to develop an algorithm and then the appropriate simulation code, though one fairly straightforward method would be to allow for random velocities that change at each collision. A realistic and intelligent object in this environment, such as a commercial aircraft realizing it is too close to another aircraft, or a tank realizing it can fire at a ground target, would possibly provide great research insight. Constant improvements in simulation and computer hardware make the complexity of adding these capabilities a plausible reality in the future.

5.4. Distributed simulation and portability

The use of sectoring in a distributed environment appears to be plausible and beneficial. A similar methodology was developed and utilized by a group of researchers modeling moving "pucks" in two dimensions [6]. They partitioned the simulation area into designated sectors using several computer processors in parallel, with each processor assigned a particular rectangular region of the trajectory space. They had similar success through an increase in computational efficiency, but had overhead in that a processor had to be assigned to provide communication between sectors. Therefore, sectors seem like a reasonable approach for distributed simulation, where different simulations (or simulators) are linked and must exchange real-time information and data.

Portability is a particular benefit of this simulation code. Written in MODSIM II, it is compilable on both a Windows-based personal computer, as well as any UNIX system. The code is available from the author to any interested user.

References

[1] A. Law, W.D. Kelton, Simulation Modeling and Analysis, McGraw-Hill, New York, 1991.
[2] MODSIM II Reference Manual, CACI Products Company. La Jolla, CA, 1990.

[3] R.H. Myers, D.C. Montgomery, Response Surface Methodology, Wiley, New York, 1995.
[4] J.O. Rawlings, Applied Regression Analysis: A Research Tool, Wadsworth, Pacific Grove, CA, 1988.
[5] E. Paulo, L. Malone, Mathematical model of sectoring in 3D space, Mathematics and Computers in Simulation, Elsevier, Amsterdam, 1999, pp. 285-296.
[6] W. Bishop, M. Hembruch, A Transputer Based Simulation of Colliding Pucks, CCECE/CCGEI '95, 1995, pp. 920–924.
[7] J.G. Cleary, Colliding pucks solved using a temporal logic, Proceedings of the SCS Multiconference on Distributed Simulation 22 (1) (1990) 219–224.
[8] C. Doescher, A statistical verification of the improvement in computational efficiency of uniformly distributed spatial objects using fixed sectoring simulation, Master's thesis (unpublished), University of Central Florida, Orlando, 1996.
[9] A.P. Goldberg, Object-oriented simulation of pool ball motion, Master's thesis (unpublished), University of California, Los Angeles, 1984.
[10] G.J. Harless, Improving computational efficiency in the discrete-event simulation of non-uniformly distributed autonomous spatial objects, Doctoral dissertation (unpublished), University of Central Florida, Orlando, 1995.