1-1-1993

Intelligent Placement Of Disaggregated Entities

Robert W. Franceschini

INTELLIGENT PLACEMENT OF DISAGGREGATED ENTITIES

ROBERT W. FRANCESCHINI

INSTITUTE FOR SIMULATION AND TRAINING

Intelligent Placement of Disaggregated Entities

Robert W. Franceschini
Institute for Simulation and Training
12424 Research Parkway, Suite 300, Orlando FL 32826

Student Paper

1 Abstract

Battlefield simulations may be classified according to the granularity of the simulated entities: military units, such as companies and platoons, may be modeled as atomic entities or as collections of vehicles. Linking a simulation system which treats units as atomic with a simulation that represents individual vehicles produces a simulation with advantages of both methods. However, this linkage requires the individual vehicles of the units to be instantiated during the simulation; these vehicles must be initially placed in a formation. This paper describes a method for placing vehicles which considers the formation template, tactical constraints and the specifics of the terrain.

2 Background

Battlefield simulators have traditionally been constructed using one of two philosophies: *aggregate simulation* or *entity-level simulation*. Aggregate simulators model companies as atomic units while abstracting the details of the vehicles which make up the companies. Battles between companies are resolved using stochastic equations. One such simulator is the Eagle simulator, developed by the US Army TRADOC Analysis Command (TRAC).

In contrast, entity-level simulators model individual vehicles, usually without modeling the hierarchical structure of platoons, companies, etc. Battles are resolved by considering vehicle dynamics, locations, and sighting abilities. SIMNET provides an environment for such simulators [Pope,1991] [Thorpe,1987]. One example of such a simulator is the Computer Generated Forces (CGF) Testbed, developed by the Institute for Simulation and Training (IST) [Smith,1992].

More information on aggregate and entity-level simulators can be found in [Mastaglio,1991] which refers to aggregate simulators as "computer-supported wargame simulations" and entity-level simulators as "networked simulators".

2.1 The Integrated Eagle/BDS-D Project

To increase the realism of each of these types of simulation, research teams at TRAC, IST, Los Alamos National Laboratory, and the Naval Postgraduate School have proposed a scheme whereby two simulators using different entity granularities to model military scenarios would interoperate during the same scenario [Karr,1992]. This, the Integrated Eagle/BDS-D project, involves the Eagle simulator, a Simulation Integration Unit (SIU), a Personal Stealth, and several IST CGF Testbeds (for the purposes of this paper, we will focus on the Eagle simulator and the IST CGF Testbeds; further information on the other components can be found in [Karr,1992]).

In a typical Integrated Eagle/BDS-D scenario, the aggregate simulator (Eagle) tracks the position and movement of companies. At critical points in the simulation (e.g., just prior to a battle), companies are *disaggregated*, which means that their representations are transferred from the aggregate simulator to the entity-level simulators. After disaggregation, human operators take control of the entities and use the entity-level simulators (SIMNET) to resolve the conflict between opposing companies. Following the battle, the companies are *aggregated*, which means that their representations are transferred from the entity-level simulators back to the aggregate simulator. The relationship between the aggregate simulator and the entity-level simulators in the Integrated Eagle/BDS-D project is depicted in Figure 1. Multiple IST CGF Testbeds are required in order to support the large number of entities which make up the military units simulated by Eagle.

2.2 The Problem

During the disaggregation process, highly detailed vehicle models must be created for each of the entities in the company. Because the vehicles are not individually tracked in the aggregate simulator, the

wargame
"network

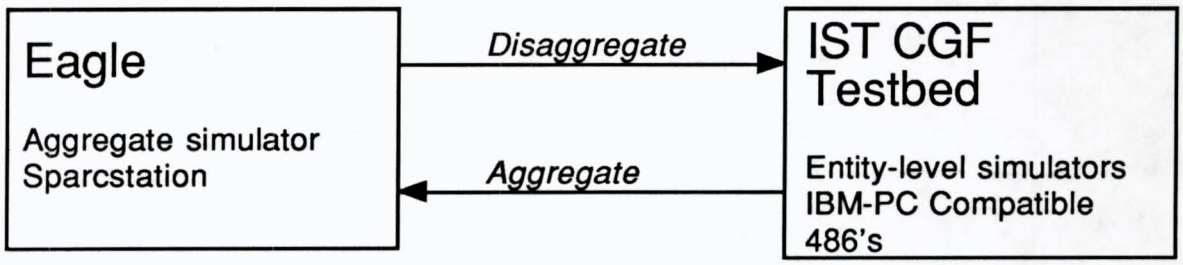

Figure 1: Disaggregation and Aggregation in the Integrated Eagle/BDS-D Project

locations of the vehicles must be decided during disaggregation and placed in these models.

The aggregate simulator provides the company's location and heading to the entity-level simulators. It also requests that the vehicles be placed in a certain formation. That formation is chosen by the aggregate simulator based on the company's current tactical situation. Formations include: Assembly, Wedge, Vee, Line, and Column. Predefined templates have been provided to the entity-level simulators for these formations. For each type of company (e.g. Tank, Mechanized Infantry), a template is defined as a set of platoon offsets from the company "center" and a set of vehicle offsets within each platoon. A simple approach to placing the newly instantiated vehicles is to simply place them on the terrain as specified by the applicable template.

The problem with simply computing offsets from the company's center is that important tactical and physical constraints are ignored. For example, entities which are in the same section (subdivision of a platoon) must be able to see each other. Further, entities which are in the same company should not be separated by an unfordable river. As a final example, entities should not be placed within physically impassible locations like rivers, lakes, or dense forests.

An alternate method to simple offset calculation for solving this problem is to place the burden on a human operator. The human operator can position the vehicles correctly using the commands available through the IST CGF Testbed. However, because a human operator will typically be required to control eight to twelve vehicles, placing these vehicles correctly in the terrain and in formation by hand takes a prohibitively long time.

Therefore, an automatic method of placing vehicles is needed which considers the information available from the aggregate simulation, the terrain database, and tactical and physical constraints. This paper presents the method that is used to solve this problem in the Integrated Eagle/BDS-D project.

2.3 The SIMNET Terrain Database

The tactical and physical constraints are ultimately expressed in terms of terrain. Therefore, the format of the terrain database used by the entity-level simulator is crucial to the processing of this algorithm.

In the SIMNET format, base terrain is represented by unordered lists of polygons, while forests, trees, and buildings are represented separately as lists of points. Base terrain includes land and water.

Each polygon in the list of polygons has a flag which tells what type of terrain the polygon represents (e.g., soil, sand, road, water). Therefore, polygons which represent the rivers, lakes, and roads of the terrain are intermixed with polygons that represent land and thus have no high-level representation. This complicates terrain reasoning since it is much harder to determine, for example, when a given polygon is part of a river or lake, or which polygons in a road are adjacent.

3 Acceptability Criteria

There are several criteria that any acceptable solution to the entity placement problem must meet.

In the Integrated Eagle/BDS-D project, the disaggregation process is allocated one minute to

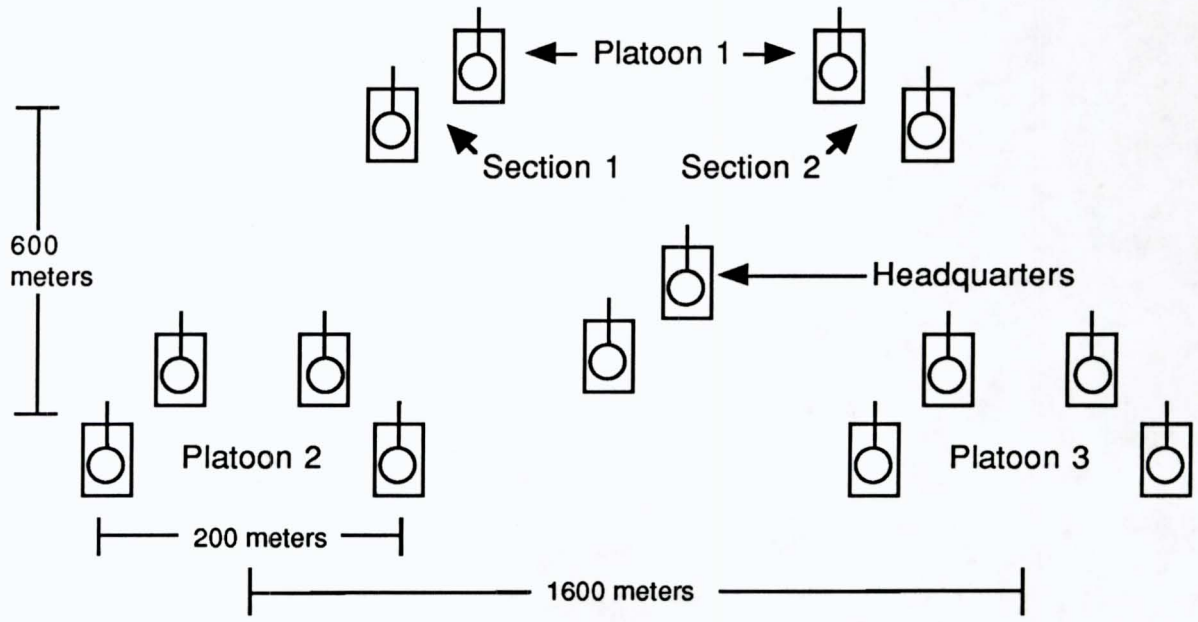

Figure 2: Wedge Template

execute. Therefore, any proposed method for placing entities in the terrain must be computationally inexpensive.

The individual entities from the companies will be placed in one of the following formations: Wedge, Line, Column, Vee, or Assembly. Templates for these formations (containing information about the location of entities relative to the company's center) are predefined and available to the IST CGF Testbed; one example template is given in Figure 2. Any proposed method for placing entities must be capable of using any of these formations as a guide for placement.

In addition, there are *tactical* and *physical constraints* which are imposed by the terrain database on the solution of the placement problem.

Tactical constraints are limits on the locations of the placed entities due to standard military doctrine. For example, a clear line of sight must exist between each section in a platoon (entities in a platoon are grouped into *sections* containing two entities--or four entities in the case of a mechanized infantry section). Another tactical constraint is that entities in the same company should not be placed across an unfordable river. Finally, entities should not be placed too close to each other, since each is expected to cover a specific amount of terrain.

Physical constraints are limits on the locations of the placed entities due to the physical interpretation of the geography of the terrain. For example, for the purposes of this simulation, entities may not be placed inside of trees, treelines, dense forests (expressed in SIMNET terrain databases as canopies), houses, rivers, or lakes.

Since there are many constraints on any method that places entities as described above, this is a difficult problem in the general case. However, for the purposes of the Integrated Eagle/BDS-D project, we assume that the terrain selected will be conducive to the selected formation. Therefore, an acceptable final result may be obtained in a reasonable amount of time by an algorithm for this problem.

4 Algorithm and Analysis

In this section, we present a heuristic-based algorithm for dealing with the entity placement problem. This algorithm has two stages: initial placement and updating.

As the first stage in the placement process, entities are placed blindly in the terrain according to previously

specified templates of the requested formation. Templates are specified using position offsets; these offsets are taken to be from the specified company's center. This initial placement uses information about the location of the company's center and the company's heading, but does not consider any information from the terrain database. Therefore, as a result of the first stage, the entities are in perfect formation, according to the template.

The second stage of the algorithm modifies the initial placement by considering information from the terrain database. This stage has a hierarchical structure which mimics the structure of a company. For the purposes of this algorithm, this hierarchical structure is described by *levels*. A level is one of the following (in order from highest to lowest): company, platoon, section, or vehicle. Checking the criteria for each level involves determining whether all conditions specified above are met by the level; if some condition is not met, the level is moved in the terrain and checked again. Levels are moved by displacing the center of mass according to one of the following strategies:

1. Move the center of the level toward the center of the next higher level (or along the company's heading in the case of the company level).
2. Move the center of the level away from the center of the next higher level
3. Move the center of the level toward the center of one sibling level (for example, on the vehicle level, two vehicles would be moved toward each other).

In summary, stage two can be described as follows:

```
function StageTwo(level) : boolean
  success = false
  tries = 0
  while (not success) and (tries < MAX_TRIES) do
    success = CheckAndUpdate(level, tries)
    if success then
      for each sublevel of level do
        success = success and StageTwo(sublevel)
    tries = tries + 1
  return(success)
```

where CheckAndUpdate performs the following operations:

1. Check that the criteria are met for *level*.
2. If any criterion is not met, then move the center of mass of *level* and try again.
3. If *level* has been moved several times already (currently, the maximum number of attempts at success is three), then report failure. In this case, the previous level's placement fails and an attempt is made to move it and try again, using a different updating strategy.

We would like to be able to analyze this algorithm to determine how its running time depends on the number of entities in a company, n. However, analyzing the complexity of this algorithm in terms of this n is complicated by the fact that terrain reasoning is involved. The complexity of terrain reasoning routines is typically measured in terms of the number of polygons processed from the terrain database as opposed to the number of entities involved in one computation. For example, calculating the line of sight between two points does not depend as much on the fact that there are two points as it does on the number of polygons between those two points. For the purposes of this analysis, the calculations are simplified by assuming that the terrain reasoning computations are independent of n and therefore occur in constant time (even though these algorithms are non-trivial in reality).

Recall that the constraints of the problem require that there be a clear line of sight between entities in the same section. In practice, this means that there is one line of sight check per section per try (hence at most one check for every two vehicles per try). Therefore, the contribution of the line of sight checks to the complexity of the algorithm is of order n. All other computations for checking criteria occur at most a constant number of times per entity (the worst case constant is on the order of MAX_TRIES raised to the power of the number of levels), and there are a constant number of these criteria checked for each entity. Therefore, because of the hierarchical structure of this algorithm, it is of order n, where n is the number of entities in the company.

5 Implementation

The intelligent placement algorithm was implemented in C++ on an IBM-PC Compatible 486 running at 50 MHz; the code is contained within the CGF Unit Supervisor module of the IST CGF Testbed. This section describes the implementation of the criteria

checking parts of the algorithm. The methods used to check for the four criteria discussed in this section are representative of the methods used for the remaining criteria.

5.1 Line of Sight

The IST CGF Testbed includes a facility for determining whether the line of sight (LOS) between two entities is clear; the algorithm used is described in [Smith,1992] and is analyzed in [Petty,1992]. Briefly, a version of Bresenham's line algorithm [Foley,1982] is used to trace polygons in the terrain database along the line of sight between two points and to determine whether terrain polygons, treelines, tree canopies, or objects block the line of sight. In the implementation of the algorithm described in this paper, the LOS computation facility is used to ensure that no LOS obstacles are between any entities in the same section. This check is implemented on the section level within CheckAndUpdate and is simply a function call to the line of sight routines.

5.2 Checking for Entities Separated by Unfordable River

Recall that in the SIMNET format terrain database, there is no high-level representation of rivers or lakes. This means that any processing which determines whether entities are on the same side of unfordable water is complicated by the fact that the shape and dimensions of the water are not conveniently available. However, the IST CGF Testbed includes a facility for planning routes between two points which considers obstacles to movement in the terrain [Smith,1992]; unfordable water is interpreted as an obstacle in this facility. To check whether two entities are on the same side of an unfordable river, the CheckAndUpdate function asks the route planner to determine a route between the two entities. If no route is available, CheckAndUpdate infers that the entities are on opposite sides of an unfordable river, and attempts to correct this problem.

5.3 Checking for Entities Within Dense Forests

SIMNET terrain is organized into 500 meter by 500 meter *patches*. Within each patch, dense forests (canopies) are unordered. Dense forests are represented in SIMNET terrain databases in two parts: a list of points comprising the boundary of the canopy and an unordered list of polygons representing the heights inside the forest. In order to determine whether an entity is within a dense forest, the CheckAndUpdate function first determines the terrain patch corresponding to the entity's location. For each canopy in the patch, CheckAndUpdate determines whether the entity's location is inside of any polygon in the canopy's representation. The number of polygons in any canopy under consideration by this project is small enough (less

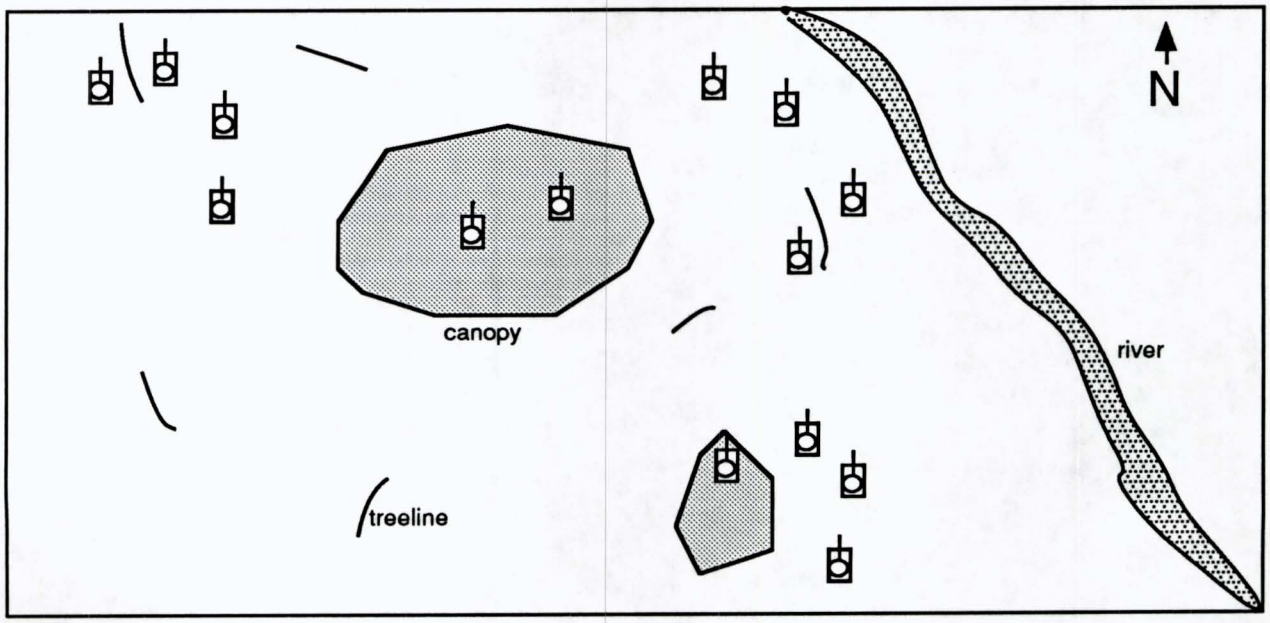

Figure 3: Wedge Formation, Without Considering Terrain

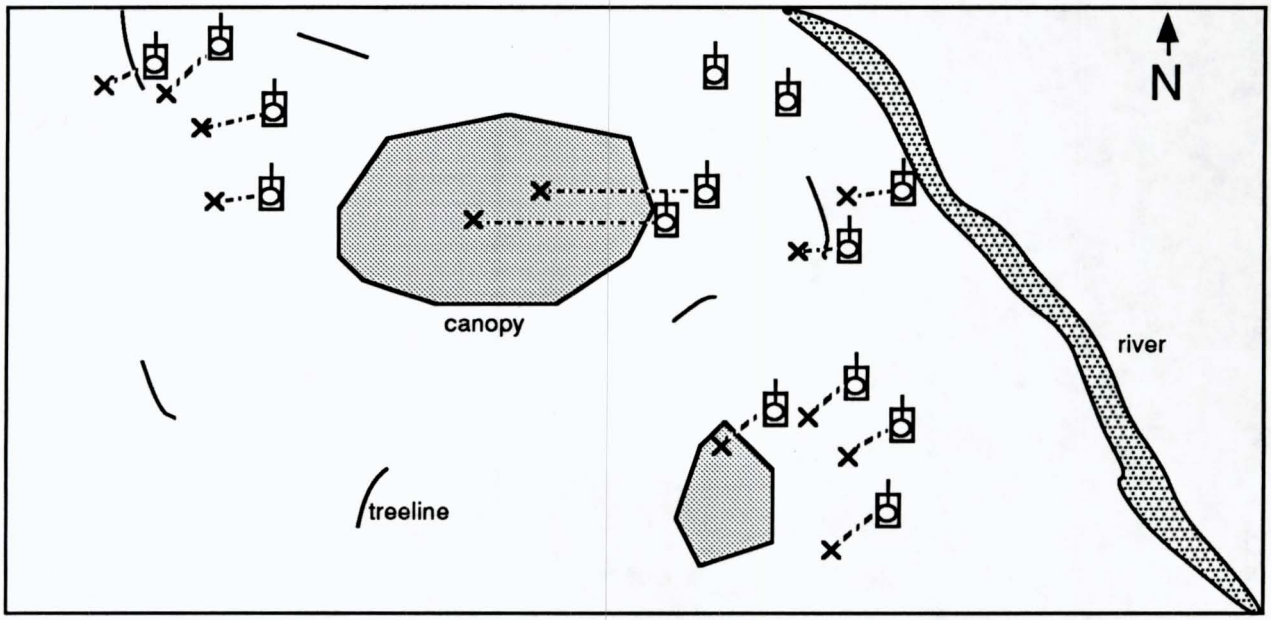

Figure 4: Positions Resulting from Algorithm

than 50) that this enumeration does not slow processing significantly.

5.4 Checking for Entities Within Water

As noted previously, water is represented in SIMNET terrain databases as polygons intermixed with land and road polygons. Therefore, to determine whether an entity is in water, CheckAndUpdate determines whether the polygon which contains the entity is a water polygon.

6 Results

An example of the results is exhibited in Figures 3 and 4. Figure 3 displays a wedge formation with heading 45 degrees from north. The placement of vehicles for this formation does not consider the terrain database; notice that tanks and infantry units were placed inside of tree canopies, and that certain vehicles in the same section do not have a clear line of sight. This is equivalent to the results of stage one of the algorithm from this paper for this disaggregation.

Figure 4 displays the results of the intelligent placement algorithm on the same disaggregation request. There are several things to notice about this placement. First, the entire company was shifted along the 45 degree heading in order to move entities out of canopies. Second, no entities were created inside canopies or treelines. Third, all vehicles which are in the same section have an unobstructed line of sight between them.

This placement took approximately 20 seconds to execute on an IBM-PC compatible 486 machine (running at 50 MHz). As discussed earlier, the time taken for this implementation depends on the size of the company and the terrain.

7 Conclusions

After the disaggregation process occurs in the Integrated Eagle/BDS-D scenario, human operators take control of the placed entities. It is important that these operators be able to give mission instructions to the troops they command immediately rather than spend time fixing the formation. The intelligent placement algorithm's results have demonstrated that the placement of the vehicles realistically in formation can be automated in a short amount of time with realism. For the purposes of the Integrated Eagle/BDS-D Project, this heuristic method of placing entities is acceptable.

8 Acknowledgement

This research was sponsored by the US Army Simulation, Training, and Instrumentation Command as part of the Integrated Eagle/BDS-D project, contract number N61339-92-K-0002. That support is gratefully acknowledged.

9 References

Foley, J.D., and Van Dam, A. (1982). *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading MA, 664 pages.

Karr, C.R., Franceschini, R.W., Perumalla, K.R.S., and Petty, M.D. (1992). "Integrating Battlefield Simulations of Different Granularity". *Proceedings of the 1992 Southeastern Simulation Conference*, Pensacola FL, Oct. 22-23 1992 (in progress).

Mastaglio, T.W. (1991). "Networked Simulators and Computer-Supported Wargame Simulations", *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville VA, Oct. 13-16 1991, Vol. 1, pp. 303-307.

Petty, M.D., Campbell, C.E., Franceschini, R.W., and Provost, M.H. (1992). "Efficient Line of Sight Determination in Polygonal Terrain", *Proceedings of the 1992 Image VI Conference*, Phoenix AZ, July 14-17 1992, pp. 238-253.

Pope, A.R. (1991). "The SIMNET Network and Protocols", Report No. 7102, BBN Systems and Technologies, July 1989, 160 pages.

Smith, S.H., Karr, C.R., Petty, M.D., Franceschini, R.W., and Watkins, J.E. (1992). "The IST Semi-Automated Forces Testbed", Technical Report IST-TR-92-7, Intelligent Simulated Forces Laboratory, Institute for Simulation and Training, February 28 1992.

Thorpe, J.A. (1987). "The New Technology of Large Scale Simulator Networking: Implications for Mastering the Art of Warfighting", *Proceedings of the 9th Interservice/Industry Training Systems Conference*, Orlando FL, November 30-December 2 1987, pp. 492-501.

10 Author's Biography

Robert W. Franceschini is a Graduate Research Assistant for the Intelligent Simulated Forces Project at the Institute for Simulation and Training. He has earned a B.S. in Computer Science from the University of Central Florida; he is currently working on both a B.S. in Mathematics and a M.S. in Computer Science at UCF. His research interests are in the areas of simulation, artificial intelligence, and graph theory.