VISUALIZATION OF ANT PHEROMONE BASED PATH FOLLOWING

A Thesis

by

Benjamin Sutherland

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2009

Major Subject: Visualization Sciences

VISUALIZATION OF ANT PHEROMONE BASED PATH FOLLOWING

A Thesis

by

Benjamin Sutherland

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved By:

Chair of Committee,	Frederic Parke
Committee Members,	Philip Galanter
	John Keyser
Head of Department,	Tim McLaughlin

December 2009

Major Subject: Visualization Sciences

ABSTRACT

Visualization of Ant Pheromone Based Path Following. (December 2009)
Benjamin Sutherland, B.S., Texas A&M University
Chair of Advisory Committee: Dr. Frederic Parke

This thesis develops a simulation and visualization of a path finding algorithm based on ant pheromone paths created in 3D space. The simulation is useful as a demonstration of a heuristic approach to NP-complete problems and as an educational tool for demonstrating how ant colonies gather food. An interactive real time 3D visualization is built on top of the simulation. A graphical user interface layer allows user interaction with the simulation and visualization.

ACKNOWLEDGMENTS

My parents' support, encouragement, and good-natured harassment have been instrumental to my development, both academically and as a person. I cannot thank them enough.

Dr. Parke has given me insight about this project as well as graphics, simulations, and writing. His technical knowledge, writing skills, and high standards have made a large positive impact on this project.

Dr. Keyser has been a positive influence on me for many years. His infectious enthusiasm and unshakeable optimism have helped keep me on track, from my undergraduate years through this project.

Phil Galanter took my initial interest in emergent behavior and helped mold it into a thesis project. He helped show me how I should think of the project, both as a task to complete and from a big-picture standpoint.

Steve Johnson of the Immersive Visualization Center helped me refine my eye for scientific visualization and gain appreciation for stereoscopy.

The people and resources of the Department of Visualization and the VizLab have been critical to my recent work and a source of joy and inspiration over the years.

TABLE OF CONTENTS

Page

INTRODUCTION 1

Problem Statement 2
Significance 2

PRIOR WORK 3

GOALS AND OBJECTIVES 7

METHOD 9

Ants 9
Food Sources 9
Pheromone Paths 10
Obstacles 12

IMPLEMENTATION 13

Ants 13
Nests 15
Pheromone Paths 15
Food Sources 18
Obstacles 19
Initialization File 19
Visualization 21
Graphical User Interface (GUI) 22
Stereoscopic Display 24

PROBLEMS ENCOUNTERED 25

Pheromone Discretization Problems 25
Path Backblow 28
Object Collisions 28

RESULTS AND CONCLUSIONS 32

FUTURE WORK 35

REFERENCES 37

VITA 39

LIST OF FIGURES

Page

Figure 1: The Surface and Map views of SimAnt, along with two of its Nest Control screens (lower left, lower center) and a "death" animation (center). (1991, Maxis Software)..... 4

Figure 2: A NetLogo-based pheromone simulation. 5

Figure 3: Two pheromone paths are shown, one longer than the other. The two pink lines are the same length, visually demonstrating that shorter pheromone paths allow for faster transit between nest and food than longer paths. This will cause shorter paths to strengthen and longer paths to fade away over time. 11

Figure 4: From top left to lower right: A. Nest (red square), food (green circle), and obstacle (black ring) at simulation start. B. Ants (pink dots) exit the nest searching for food. C. An ant finds food, heads back towards its nest as directly as it can based on its path integration, laying down a pheromone path (grey line). D. Other ants join the ant on its path, while one ant has strayed, making its own path. E. The newer path turns out to be the shorter path, causing it to be strengthened by the ants while the longer path becomes less popular. F. The shortest path emerges, and becomes strong enough that no ants will deviate from it. 12

Figure 5: Various ant history trails (solid lines) shown along with pheromone paths (dots). 15

Figure 6: From top to bottom: A: The nest (left), food (right), and pheromone path between the two. Thickness denotes strength of pheromones. B: An ant emerges from its nest, senses all around itself (red circle) for the greatest pheromone concentration, and begins moving in that direction. C: The ant moves along the path, ignoring the pheromones it has passed as it decides where to go. 17

Figure 7: An example input file. Lines beginning with "#" are comments. 20

Figure 8: The program's main window, showing a view of the 3D ant world on the right and interactive controls on the left..... 22

Figure 9: Obstacles may be moved through mouse interaction. The user clicks-and-holds on the object and drags it from its initial position (left) to its desired location (right). 23

Page

Figure 10: The menu bar, its submenus, and each action's corresponding key command. (On Linux and Windows systems, the control key is the modifier instead of the command key)..... 24

Figure 11: Path creation behavior was dependent on the direction ants were traveling..... 25

Figure 12: Pheromone points are sparse (left) before fixing the problem (right). 26

Figure 13: The pheromone path from nest to food source with the “path snapping” problem unresolved, with a solid line showing the optimal path, which is taken once this problem was resolved..... 27

Figure 14: The path (grey squares) blows back and grows past the nest (red square). 28

Figure 15: Ants were unable to cross sharp polygonal edges. 29

Figure 16: Ants would collide with a polygon face and turn around. 30

Figure 17: More objects obstructing the ants' path make for more interesting pathfinding..... 32

INTRODUCTION

Many classical computer science problems are classified as NP-complete, which means that no known method short of computing and comparing all possible solutions is guaranteed to find the optimal solution. Exhaustive comparisons can be computationally expensive and not feasible to perform. Consequently, heuristic algorithms have been developed for many of these tasks to find near-optimal solutions. Many heuristic approaches have been created using aspects of natural systems as a basis. One such basis involves ant behavior.

Ants sustain their nest through constant food scouting and retrieval expeditions. These expeditions involve older, less valuable workers scouting the terrain near their nest for food. Once food is found, the worker walks back to the nest using as direct a route as possible. This is done using a combination of path integration (a.k.a. dead reckoning) and cues from the sun's position [Wehner 2003]. On these return trips to their nest, the successful ants lay down a path of pheromones, or chemical signals, which attract other workers to follow this path. As other ants return from their gathering, they lay down pheromone paths as well. While ants will likely follow an existing pheromone path, they are not guaranteed to do so, and are less likely to do so if the path strays too far from the ant's own sense of dead reckoning. In the long term, this behavior chemically "highlights" the shortest path towards the goal, since ants traversing a shorter path will lay down a stronger cumulative pheromone path over time than those traversing a longer path. This behavior, combined with pheromones evaporating over time, allows the most successful path to a goal to become more attractive while other paths become less attractive.

The project described in this thesis was to develop and visualize an algorithm based on an abstraction of ant path finding, creation, and following (henceforth referred to as "pathing"). Users may experiment with the algorithm and its visualization through the

---

This thesis follows the style of *ACM Transactions on Graphics.*

manipulation of certain variables, interactively learning about the relationship between the variables and the algorithm's behavior.

Problem Statement

This project is an abstracted three-dimensional simulation and visualization of goal-based path finding, creation, and following based on ant behavior. This simulation's visual layer is intended to be appealing and informative. The simulation and visualization are interactive and run on Windows, OS X, and Linux systems.

Significance

The insights gained from abstractions and adaptations of ants' ability to organize and efficiently gather food can be useful for a number of human applications, mostly in areas of finding a shortest path or adding error-correction to existing algorithms. Ant pathing ideas have led to heuristic algorithms for several NP-complete problems, including the Traveling Salesman Problem [Dorigoa and Gambardellab 1997], Job Shop Scheduling [Zhang et al. 2006], and communications routing [Di Caro and Dorgio 1998].

This project seeks to facilitate understanding of the pathing basis of these algorithms through visual demonstration. The visualization gives real-time visual feedback regarding the effects of changing variables in the simulation.

This visual simulation could also be used as a teaching tool showing how ants interact with their environment and how they work together as a group to solve a shared problem. The program may also show how humans can benefit from understanding and adapting the ants' behavior in ways like those described above.

PRIOR WORK

Ants' influence has been felt since ancient times. Ants are referenced in ancient scripts such as the Talmud (Proverbs 6:6), the works of the ancient philosopher Pliny, and the Roman writer Claudius Aelianus. These references are mostly in relation to their tireless work gathering food and returning it to their nest. Formalized study of ants seems to have begun in the eighteenth century when William Gould published the first recognized myrmecological paper [Gould 1747]. Ants' use of pheromones to guide their pathing behavior was discovered by E. O. Wilson in 1958 [Hölldobler and Wilson 1990]. In the 1990s Jean-Louis Deneubourg showed that the pathing behavior of ants could be used to find an approximation of the shortest path between two points [Beckers et al. 1994].

Deneubourg's insight paved the way for computational analysis of ant pathing. Dorigoa and Gambardellab adapted ant pathing to find an algorithm for the Traveling Salesman Problem (TSP) [Dorigoa and Gambardellab 1997]. Instead of allowing the ants full freedom of movement, ants were only allowed to follow lines between cities, and ants could only visit cities they had not visited before. Over time, a near-optimal path emerges. Using the TSP abstraction of ant pathing as a basis, an ant-based Job Shop Scheduling algorithm [Zhang et al. 2006] was developed at Sun Yat-Sen University. The algorithm works in a manner similar to the TSP, with cities replaced by jobs and distance by time. Researchers from the University of Brussels worked on creating robust, error-correcting communications routing based on ant pathing [Di Caro and Dorgio 1998], a specialization of the ant TSP.

SimAnt [Wright and McCormick 1991] is a game based on ant behaviors. Players control a single ant in an ant colony, searching for food and battling other nests' ants. The player can also control some basic colony attributes like reproduction rate and total number of ants in the colony, as in Figure 1. Players can switch which ant they control, and manually lay down basic pheromone paths. SimAnt is not a true simulation, however, as most nest activity requires at least some user interaction.

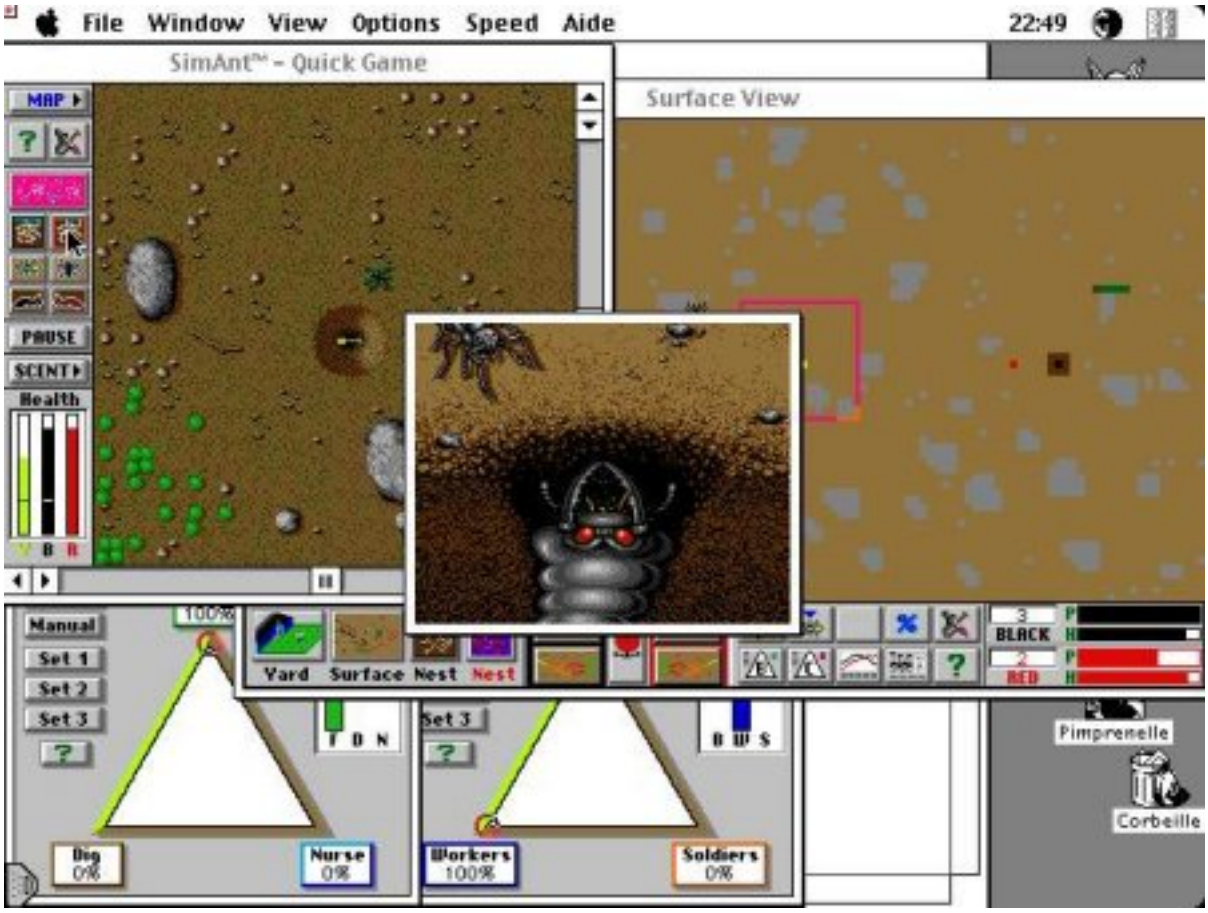

Figure 1: *The Surface and Map views of SimAnt, along with two of its Nest Control screens (lower left, lower center) and a "death" animation (center). (1991, Maxis Software)*

MacLennan developed a rudimentary simulation of ant pheromone behavior modeled using the simulation engine NetLogo [MacLennan 2008], as shown in Figure 2. This simulation is based on the Resnick model of simulated ant behavior [Resnick 1994; Wilensky 1998]. In this simulation, ants operate on a low-resolution square board and display some pheromone following behavior. Basic obstacle-avoidance is also included. Unfortunately, the ants in this simulation don't behave very well unless they are close to the nest, limiting the simulation's effectiveness. Furthermore, it makes use of a "nest scent" pheromone to guide the ants back to the nest, following the Resnick model

[MacLennan 2008; Resnick 1994; Wilensky 1998]. This phenomenon does not naturally occur. Still, this simulation demonstrates the basic principles well.

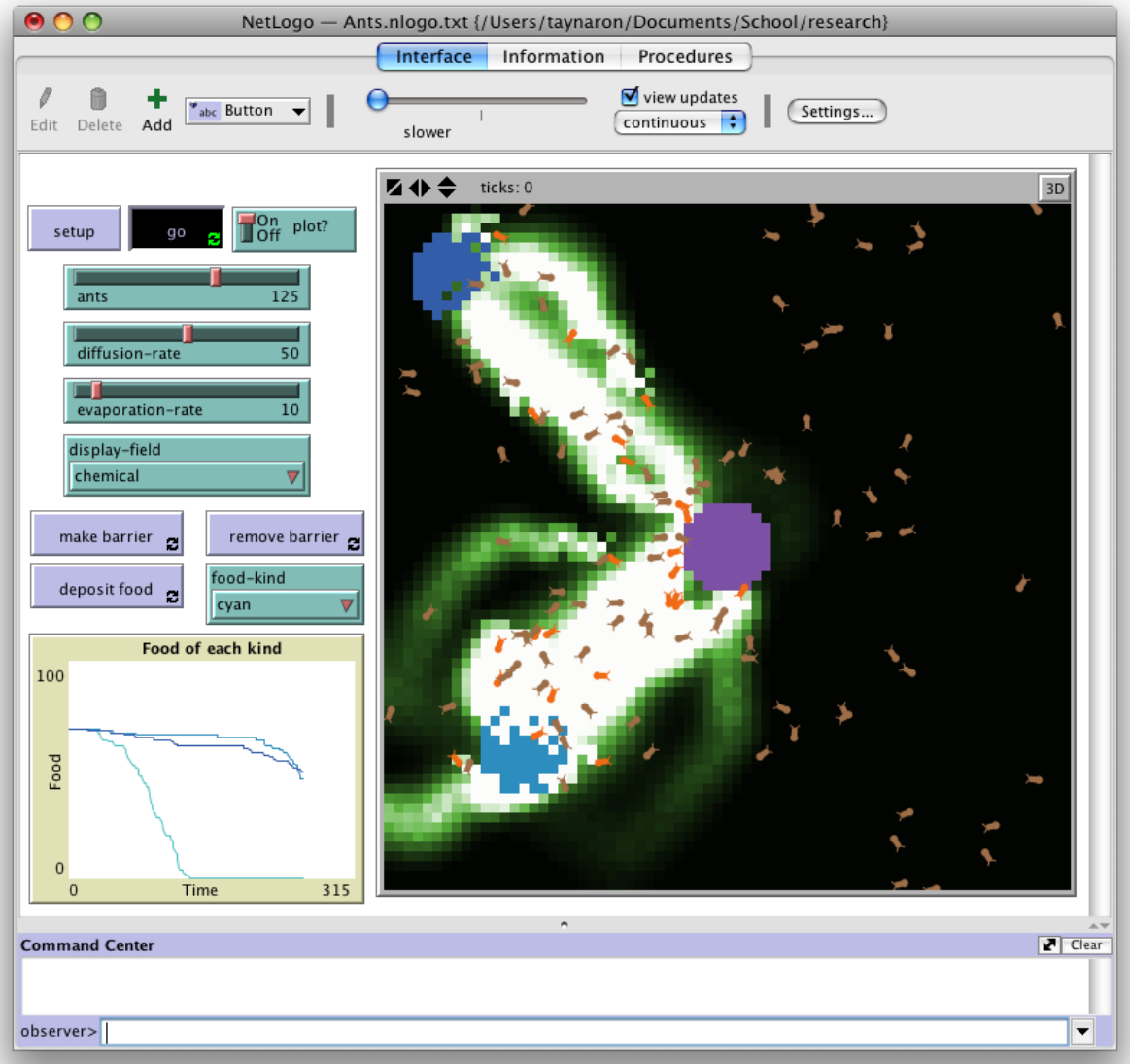

Figure 2: *A NetLogo-based pheromone simulation.*

Tatsuya Nakamura created the framework for a 3D interactive video game based around the idea of ants as a mafia family [Nakamura 2006]. The player operates as a member of this "ant mob", building up the family's fortune throughout the game. As part of this

game, non-playable character ants exhibit simulated pathing behaviors. These ants forage for food and lay down pheromone paths in a greatly simplified, simulation of real ants' behavior. Pheromone simulation is not a critical part of this game, so the simulation is not very realistic, relying on some pre-processing of each game level and storing information to which ants would not normally have access.

More recent studies on the ants themselves include studies on ants' ability to find their nest through path integration [Wehner 2003], even in three dimensions [Grah et al. 2005]. Some preliminary work has been done to create a simulation of ant pathing using miniature robots as ant surrogates [Beckers et al. 1994].

GOALS AND OBJECTIVES

To judge the success of this project, some measurable objectives are needed. These objectives should both clarify the expectations of the project and allow for an objective analysis of the successful achievement of the project's intended goals.

This simulation is not meant to explicitly mimic ant behavior or any natural process, so natural limitations such as the ground plane are not included. The simulation and visualization are extended into a three-dimensional space instead of being limited to a terrain. The 3D nature of the simulation allows for a less restricted simulation environment. The resulting visualization is more informative and visually interesting than a 2D visualization would be.

For this project to be considered satisfactory, it must meet the following objectives:

- The simulation must depict pheromone-based path creation and optimizing behavior similar to that exhibited by many ant species.
- The simulated ants must be able to:
    - Walk semi-randomly away from their nest to find food
    - Recognize and acquire food
    - Lay a pheromone path from a food source back to their nest
    - Determine the location of their nest based on path integration
    - Follow existing paths
    - Avoid obstacles
- The simulation's results should be displayed graphically in a manner that depicts the simulation in an intuitive, visually appealing way.

- The visualization and simulation should react to user input in real-time.
- The visualization should be able to be displayed stereoscopically on systems with stereoscopic capability.
- Through the graphical user interface (GUI), users should be able to:
    - Control their viewing position and angle, allowing them to see areas of interest
    - Turn on visual elements of interest while disabling visual elements of less interest
    - Interact with simulation parameters in real-time, such as pheromone evaporation rate
    - Change visual attributes such as ant history trail length
    - Add objects, food sources, and pheromone paths at any time
    - Manipulate the location of obstacles in 3D space via mouse control
    - Save and restore the simulation state
- The file created by saving the simulation state must be human readable and should be easily editable.
- The application created must be cross-platform, able to work on Windows, Linux, and OS X systems.

Please note that some of these requirements, such as visual appeal and ease-of-use, are not objective and could only be verified through user studies. These goals will not be tested as user studies are beyond the scope of this project. Still, a concerted effort must be made to achieve them.

METHOD

The pathing simulation developed is not of any particular ant species, or of real ants at all. Rather, the simulation is of an algorithm based on an abstracted aspect of ant behavior. Real world ant behaviors and the simulated approximations are described in the sections below.

Ants

Real ants may move along any surface they can hold onto, a sort of constrained 3D space. In this simulation, however, ants are not constrained to walking on surfaces. The simulated ants may be thought of as "actors" in 3D space with full 3D motion capabilities having the desire to find food for their nest.

The simulated ants do not have any memory beyond their previous few steps and a general understanding of their nest's location. Neither do the ants have any memory or understanding of the simulated world space beyond their immediate surroundings and the information they receive from pheromones [Beckers et al. 1992]. In the absence of a pheromone path, each ant operates independently, choosing a direction away from the nest and wandering away from it searching for food. Eventually, the ant will return to the nest if it has not found food, and search again. Ants do not share any information regarding unsuccessful searches, so other ants may duplicate previous searches.

Food Sources

An ant's primary objective in this simulation is to find food and return it to the nest as quickly as possible. Like real ants, these simulated ants are able to sense food from a small distance away. The larger the food source, the greater its attraction strength, and therefore the farther away the ants can be and still sense the food. Once the ants sense the food, they will walk to it, take some, and begin walking back to the nest.

Pheromone Paths

Depending on their species, real ants can lay down 10 to 20 different pheromones – chemical behavioral indicators. Pheromones are most ant species' primary form of communication. Different pheromones in varying amounts relay different signals for ants, such as following a path, avoidance, panic, and aggression [Hölldobler and Wilson 1990]. For the purposes of this simulation, only the pheromone indicating a path to a food source is simulated. When an ant finds a food source, it lays a sequence of chemical points in space from the food source back to the nest on as direct a route as the ant can manage. As with the simulated ant movements, all pathing is unconstrained in 3D space.

When the simulated ant finds food and no pheromone path exists, the ant will forge its own. If a path already exists, the ant will likely follow it. The ant may diverge from this path if its understanding of the nest location is sufficiently divergent from the path direction and the existing path is not particularly strong. Subsequent ants may follow either path or create their own based on the strength of the path and its relative directness, as in Figure 3. The more closely aligned a pheromone path is to the ant's understanding of where its nest is based on path integration, the more likely ants will take it. This likelihood reduces the number of ants that will take other paths or create their own. Over time, a single, nearly optimal path will emerge from these multiple paths, with the other paths fading away from disuse.

Ants may encounter an existing pheromone path while they are leaving their nest or searching for food. If an ant encounters a path when leaving the nest, it will follow the path away from the nest. If the ant encounters the path while searching for food, it will follow whichever direction is the least deviation from its previous course.

Figure 3: *Two pheromone paths are shown, one longer than the other. The two pink lines are the same length, visually demonstrating that shorter pheromone paths allow for faster transit between nest and food than longer paths. This will cause shorter paths to strengthen and longer paths to fade away over time.*

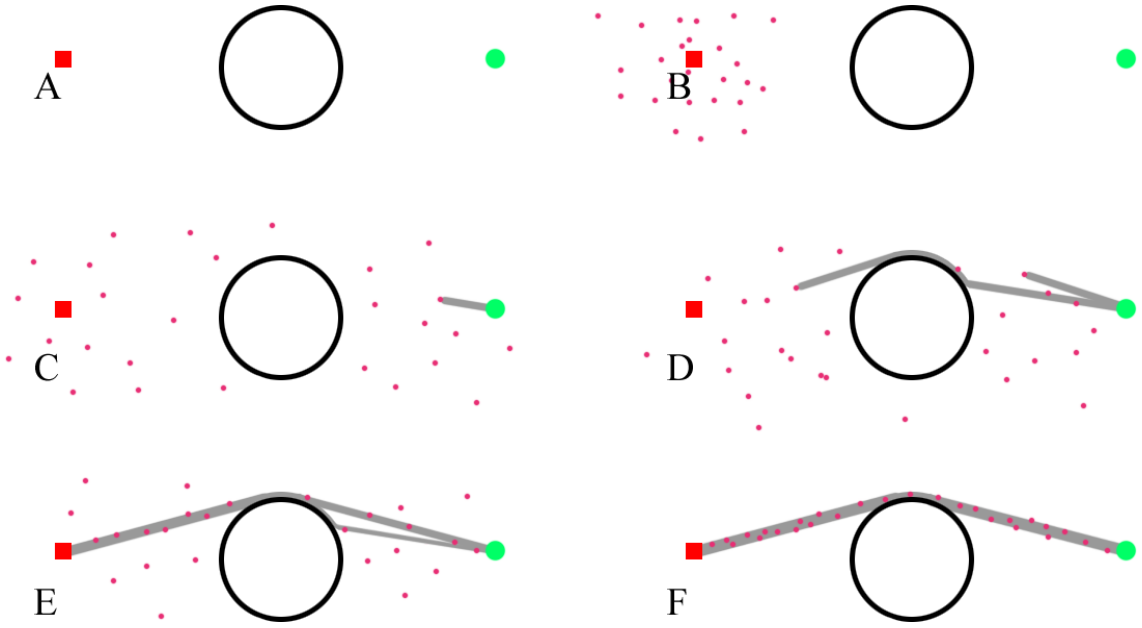

Figure 4: *From top left to lower right:*
(A) Nest (red square), food (green circle), and obstacle (black ring) at simulation start.
(B) Ants (pink dots) exit the nest searching for food.
(C) An ant finds food, heads back towards its nest as directly as it can based on its path integration, laying down a pheromone path (grey line).
(D) Other ants join the ant on its path, while one ant has strayed, making its own path.
(E) The newer path turns out to be the shorter path, causing it to be strengthened by the ants while the longer path becomes less popular.
(F) The shortest path emerges, and becomes strong enough that no ants will deviate from it.

Obstacles

Obstacles can be placed anywhere in space by the user during the simulation. These obstacles may impede the ant's movement, blocking previously established pheromone paths. Ants following a pheromone path that becomes obstructed by an object will create new paths around the object over time, eventually finding a new best path, as in Figure 4. If the obstacle is then removed, the ants will again create new paths, eventually reverting back to a more optimal path.

IMPLEMENTATION

The world described in the Method section is simulated in discrete time steps. At each time step, all objects in the simulation are updated (except those explicitly defined as having a different update schedule in this section) based on the state of the world in the previous time step. Time steps are not coupled to frame rate – the rate at which a new visual image is calculated – allowing both the simulation and visualization to run without having to wait on each other. The radius ants can explore away from their nest may be defined at the beginning of the simulation via an input file. The program then determines the size of the octree – a data structure designed to hold sparse 3D data – necessary to make an appropriately spaced pheromone grid.

Ants

Upon leaving the nest, each simulated ant determines a random starting vector. The ant moves away from the nest in a semi-random walk in that direction. Slight deviations in heading are allowed between time steps, emulating real ants' movement, which is more of a directional wander than purely random [Hölldobler and Wilson 1990]. The ant will wander until either finding a food source or reaching the defined search radius limit. In either case, the ant will endeavor to return to the nest as directly as possible. If the ant has found a food source, it will pick up food and lay down a pheromone path on its return to the nest.

Real ants use a combination of path integration, landmark recognition, and sun position to determine location relative to the nest. In this simulation, since there is no sun and few landmarks, ants use path integration to return to the nest. Path integration is essentially a summation of the total vector distance moved since leaving the nest, and will therefore always lead back to the nest. Ants in this simulation have perfect memory of their previous positions. This gives them perfect path integration in the simulation, which does not occur in real life. Some previous position jitter has been added to the simulation to better reflect real ants' behavior.

The three basic ant options (leave/return to nest, move randomly, follow path) are combined to produce an ant's behavior at any given point. The combination is achieved by combining the result of each option multiplied by a weighting factor for each. The weighting of each option was tuned iteratively, and has no basis in reality beyond matching observed ant behavior. The specific numbers used to tune the behavior mean nothing outside of the code, so the numbers will not be reproduced here. In general, however, ants are always most interested in following pheromone paths. If they have food, the ants prioritize returning to the nest over random movement, but may still make detours based on errors in their dead reckoning. If the ants have no food, they emphasize their semi-random walk over moving away from or returning to the nest.

Ant simulations based on the Resnick model [MacLennan 2008; Resnick 1994; Wilensky 1998] use a second pseudo-pheromone to help the ants find their way back to their nest. This "nest scent" is a simple way to get the ants to return to the nest, but it has no basis in reality.

The simulated ants have a maximum lifespan. When they exceed this lifespan, they die and a new ant is spawned at the nest. The lifespan is many times longer than a traversal from the nest to the ant's exploration range and back should take. This only becomes relevant in the rare instance when an ant has become hopelessly lost or stuck on or in an obstacle.

Each simulated ant may display a visual history of its last few movements as a trail. This history trail should not be confused with a pheromone path, as it has no effect on the simulation, and is a visual effect purely for the benefit of the viewer. Blue trails are exploring ants, red trails are ants returning to the nest after an unsuccessful search, and green trails are generated by ants returning with food. The user may control the length of the history trails shown or disable this function entirely.

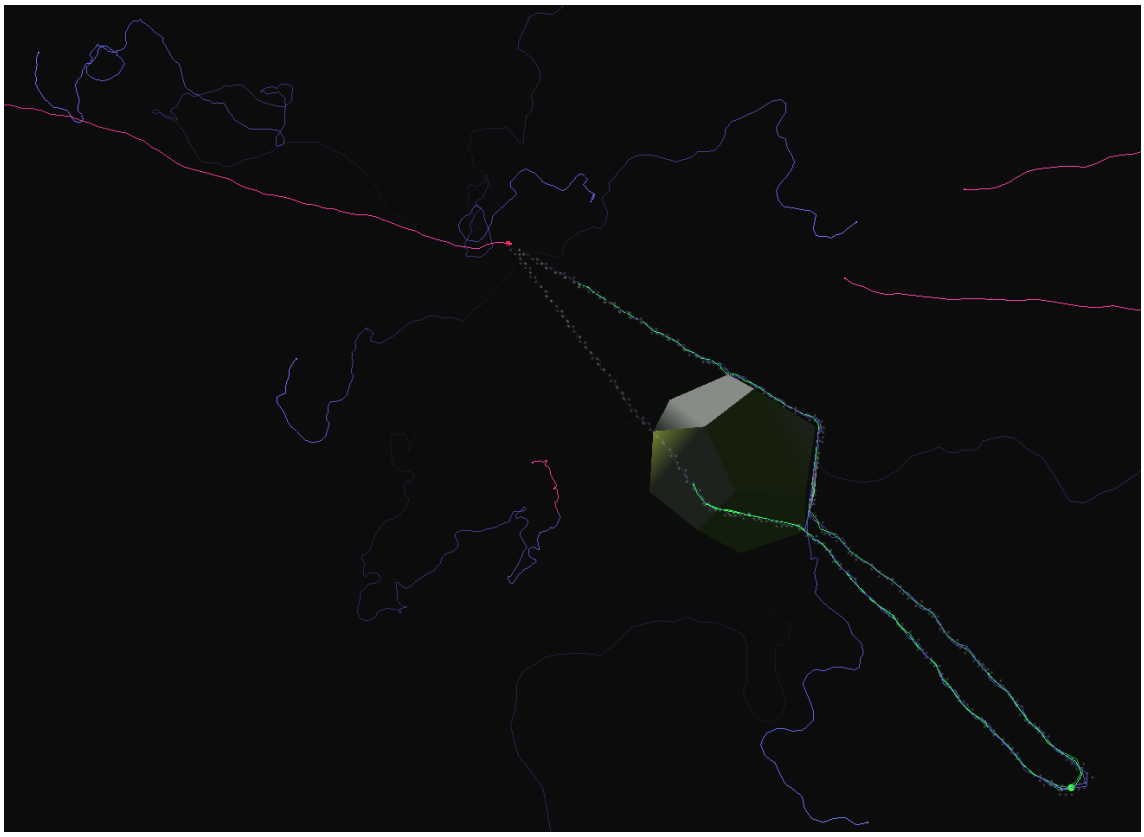

Figure 5: *Various ant history trails (solid lines) shown along with pheromone paths (dots).*

Nests

The user may define multiple nests in the world space. Each nest has its own maximum number of ants, ant spawning rate, and exploration radius. A separate worldwide ant limit also exists. Nests cannot spawn more ants if the worldwide limit has been reached, even if its own limit has not been reached. All these variables are initialized in the simulation setup file.

Ants from different nests do not interact with each other in any way, and ants will not follow another nest's pheromone paths.

Pheromone Paths

When a simulated ant has found a food source, it lays a sequence of chemical points in space from the food back to the nest on as direct a route as it can manage, as in Figure 5.

In the simulation, pheromone points are floating-point values (denoting strength) in space stored in an octree structure. Most of the space is empty, since few paths exist at any one time. Consequently, using an octree or similar structure reduces memory usage over a normal 3D array.

As they are created, pheromone points are stored at the nearest set of spatial coordinates available in the octree. For example, if the grid size is 0.1 units, something at point [10.235, 6.778, 12.901] would be stored in the position [10.2, 6.8, 12.9]. This is referred to as discretization.

Each pheromone point has a radius of attraction based on its strength. At each time step a simulated ant aggregates all the pheromone points it detects. It ignores pheromone points it has just passed. This prevents it from getting stuck on a point of high attraction. If the ant has just exited its nest, it senses around itself for pheromones. If no pheromone points are in its sensing vicinity, it continues its semi-random walk looking for food. If pheromone points are detected, the ant picks a movement vector which best fits the aggregated points. The ant will then follow this path, at each time step finding the greatest concentration of forward-facing pheromone and heading in that direction (Figure 6). If the ant has food with it, it will add its own pheromone to each of the grid points it passes through, strengthening them, as it returns to the nest. Pheromone points dissipate over time at a user-controllable rate. When a pheromone point dissipates below a certain threshold, it is culled from the octree structure.

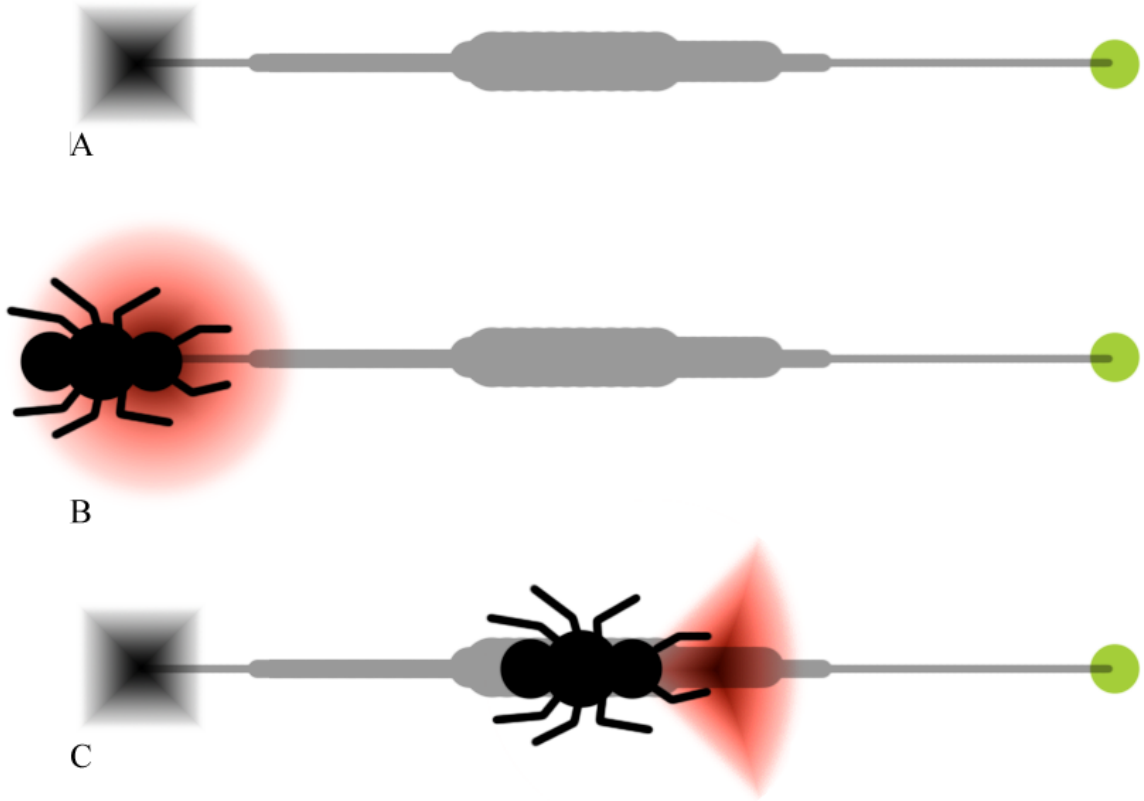

Figure 6: *From top to bottom:*
(A) The nest (left), food (right), and pheromone path between the two. Thickness denotes strength of pheromones.
(B) An ant emerges from its nest, senses all around itself (red circle) for the greatest pheromone concentration, and begins moving in that direction.
(C) The ant moves along the path, ignoring the pheromones it has passed as it decides where to go.

In the absence of an existing pheromone path, the ant makes its way to its nest as directly as possible. Like real ants, however, these simulated ants are not aware of their nest's exact location. As previously stated and in [Grah et al. 2005; Wehner 2003], real ants use path integration and cues from the sun to determine the location of their nest. This behavior is simulated by adding the ant's previous movements together to create a resultant vector between the ant's current position and its nest. A small jitter is added to each movement to simulate ant's imperfect memory. As a real ant approaches its nest, it

adjusts its heading back towards the nest based on visual cues. This is simulated by retuning the ants path memory to become more accurate as it approaches its nest.

If a pheromone path already exists near the food source, the ant's path back generally follows this path. As the ant follows this path, it lays down its own pheromones, reinforcing the path. It is possible, however, for the ant to break away from this path and attempt to find its own way. The likelihood of this action is based on the strength of the pheromone path and its perceived directness to the nest. The weaker the established path, the more likely it is that the ant will break away from the established path to create its own. Subsequent ants will choose between the existing paths or creating their own. The more direct a path is, the faster ants can traverse it. The quicker ants can traverse a path in a given time interval, the more it will be reinforced. Therefore, over time, the shortest of the paths will be the one in use.

Additional pheromone paths may be manually placed by the user by interactively defining start and end points. The program creates a line of pheromone points in between these points.

Food Sources

Each food source has a position and strength upon which its attraction radius is based. Any ant within the food's detection radius will approach and find the food. The ant then returns to its nest, laying down a pheromone path along the way. Food sources may be placed at any time during the simulation by the user and may be one of three types:

- Constant: have a constant strength, do not decay
- Timed: have an initial strength and decay over time, eventually disappearing
- Finite: have an initial strength that is diminished each time an ant contacts it, the same way a food source is depleted by ants carrying it back to their nest piece by piece.

Food sources can be thought of as goals if one wanted to further abstract this program's use for simulation of multi-node pathing, but for the purposes of the current simulation it is unnecessary to consider this abstraction.

Obstacles

The user may add obstacles at any time at any location in simulation space. Obstacles are defined as triangulated polygonal objects specified using OBJ-format files, a standard 3D object format. To guarantee efficient and accurate handling each obstacle should be a single closed-surface convex object. Other object types may work, but could lead to undesirable behavior such as ants getting stuck in minor concavities.

Obstacles are handled in one of two ways. If the obstacle is not interfering with a pheromone path, ants will occasionally bump into it or wander around, on, or near it, but will not penetrate it. If the obstacle blocks a path an ant is currently following, the ant will perform a semi-random walk along the surface of the object. Possible steps that have the smallest angle of deflection from the original path are weighted more heavily in an attempt to re-link with the path in the most efficient manner possible. If the ant cannot detect pheromone after continuing in the same direction it has been walking for a few time steps, it will then resort to dead reckoning toward its current goal. As the ant moves around this obstacle it releases a smaller amount of pheromone (if it is returning to the nest), creating a path that quickly fades away. This smaller amount will draw other ants toward the potential solution but make them more willing to try other paths, allowing them to more quickly find a near-optimal path over a short distance than with the regular pheromone amount. Once a clear path to the goal is found, either by finding an existing path or by having a clear dead-reckoning vector towards the nest, the ant resumes its previous pheromone-laying activity (laying none if it has no food, or laying the regular amount if it was headed back to the nest with food).

Initialization File

Defining the simulation world and simulation setup is managed through an initialization input file. Global variables such as the time step and the maximum number of ants

allowed may be set, as well as the placement and setting of parameters of nests, food sources, objects, ants, and pheromone points. All commands are in human-readable format, with a keyword to the left of the equal sign and associated data to the right. Initializing pheromone points by hand may be tedious given the large number of points needed to construct a path, and is not recommended. Instead, the user may load a new or previously saved scene and use the previously mentioned pheromone path creation tool, then save the results when satisfied.

```
#world parameters

Default time step = 0.002
Maximum ants = 150

Current position = 0 0 0
Ant history length = 15
Exploration radius = 120
Ant spawning rate = 1
Nest = make

Food amount = 50
Current position = 50 50 50
Food = make

Current position = -70 20 -50
Food = make

Current position = 20 20 20
Object size = 5
#Object initial rotation = 0 0 0
Load object = objs/dodecahedron.obj
```

Figure 7: *An example input file. Lines beginning with "#" are comments.*

The values associated with each parameter are remembered during file input, so similar parameters need not be repeated for multiple objects. If no value for a parameter is specified in the input file, the program will provide a default value. For example, in Figure 7 above the food amount for the first food source is set to 50, and as no food

amount is set for the second source, it defaults to 50. When all the attributes have been changed to their desired value for an object, the 'make' keyword is used to actually create the object. The user may include comment lines by beginning them with the "#" symbol. The simulation world state can be saved or restored at any time using the UI. The saved state file is in the same user-readable format as the initialization file, allowing easy editing of the state.

Visualization

Visual images of the simulation are rendered in real-time using OpenGL. These images have a minimalist, pseudo-retro computer graphics feel, as in the movie TRON [Miller and Lisberger 1982], with bright solid colors and sharp lines. Ants are represented by a single point in 3D space, with an optional "history trail", highlighting the ant's previous steps.

A pheromone path is displayed as a series of semi-transparent points in space. These points must be rendered in order of farthest to nearest in space relative to the camera because of the way OpenGL handles transparent objects. The points are therefore sorted from back to front using the octree structure. At each level of the octree, the octants (a space divided in half in each dimension, creating 8 smaller spatial areas) are sorted by distance to the camera, from greatest to least. When a node is reached, it is inserted into a doubly-linked list in order of distance. The location of the last inserted object is stored for faster insertion times, since the next object to insert is likely to be adjacent to the currently selected node. While constructing this list has a worst-case runtime of $O(n^2)$ (since it is an insertion sort), the presorting used by the octree structure allows an average runtime of $O(n \log(n))$. This list is generated every eight simulation time steps (or when the camera moves) and is cached for display. Regenerating the list in this manner was a compromise between appearance and speed. Updating at every time step slowed down the simulation considerably, and offered little visual improvement over

caching. Updating at every time step is necessary when moving the camera, lest items get out of order.

Users have the ability to turn on or off visibility of all the parts of the visualization in order to focus on specific aspects of interest. Disabling visibility of part of the visualization does not affect the underlying simulation.

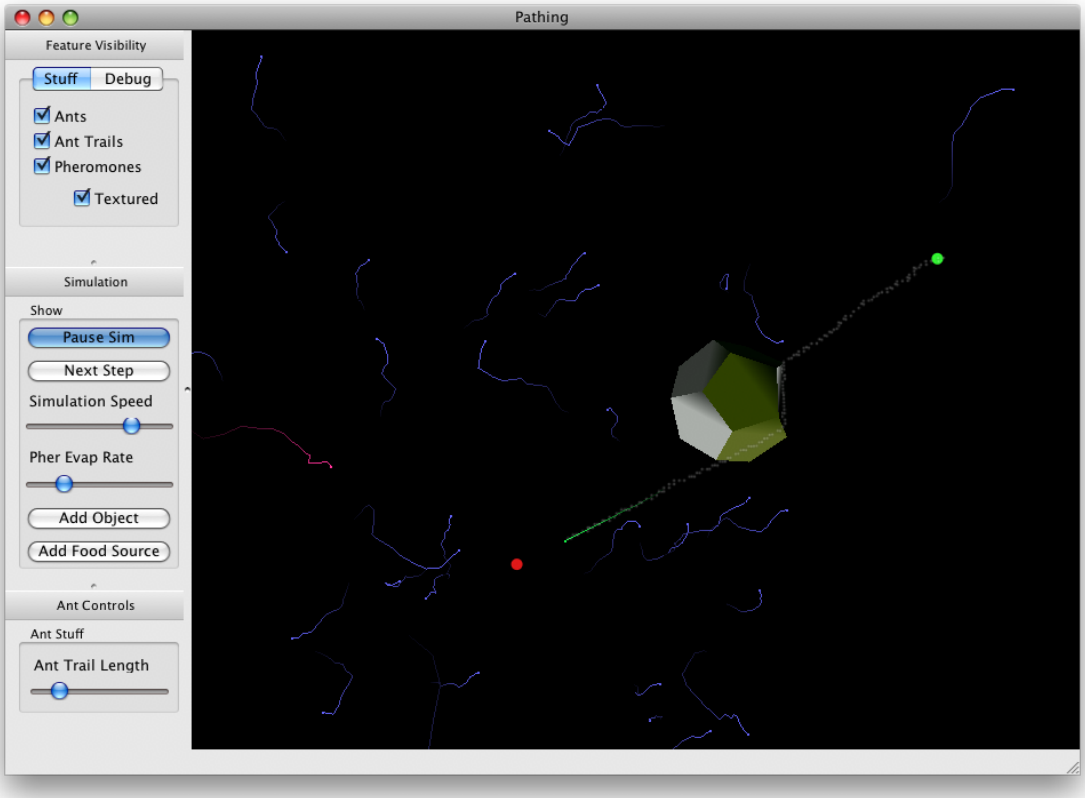

Figure 8: *The program's main window, showing a view of the 3D ant world on the right and interactive controls on the left.*

Graphical User Interface (GUI)

The GUI was created using the Qt windowing toolkit [Nokia 2008]. Qt is a cross-platform, open-source application and user interface (UI) framework. It consists of a series of libraries through which programmers can quickly and easily create an application.

The interface for this simulation is made up of three parts; the UI sidebar, the menu bar at the top of the main window (on Windows and Linux) or at the top of the primary monitor (on OS X), and mouse interaction through the simulation viewing window, as shown in Figure 8. Using mouse interaction, the user can move around in the scene, add pheromone paths, and manipulate obstacles (as in Figure 9) directly in the 3D simulation space.

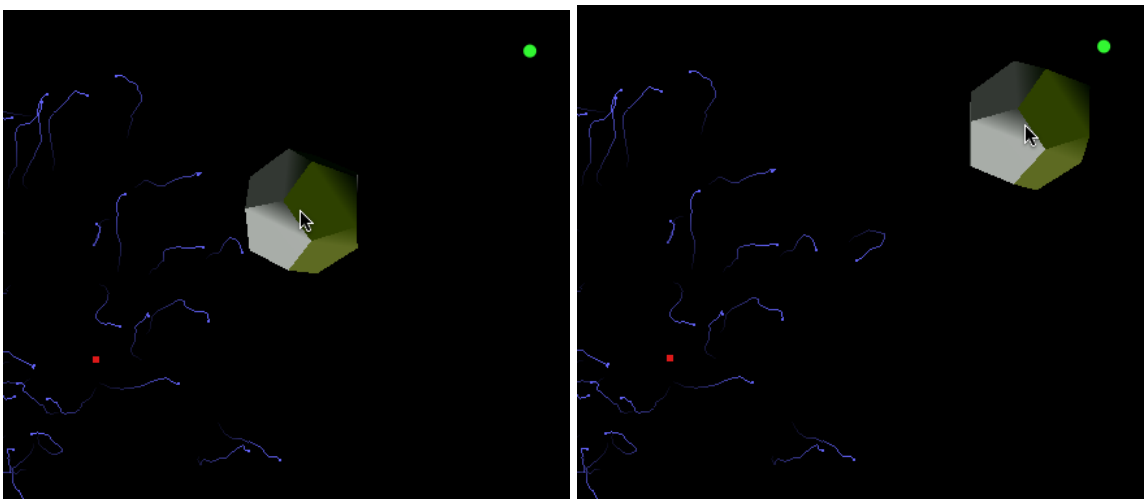

Figure 9: *Obstacles may be moved through mouse interaction. The user clicks-and-holds on the object and drags it from its initial position (left) to its desired location (right).*

Other interaction options use the UI sidebar (as shown in Figure 8), the menu bar (as shown in Figure 10), or both. From the side bar, users may toggle the visibility of aspects of the simulation, control pheromone evaporation rates, history lengths, pause the simulation, and add objects and food sources. From the menu bar, all the visibility toggles are available as well as the ability to save and load the simulation and an option to hide the sidebar, allowing the visualization panel to take up the whole window. Keyboard shortcuts are provided where applicable, as shown in Figure 10.

The mouse wheel changes the depth of the mouse action plane. All mouse interactions with objects in the simulation take place on this plane.

All inputs have immediate effect on the simulation.

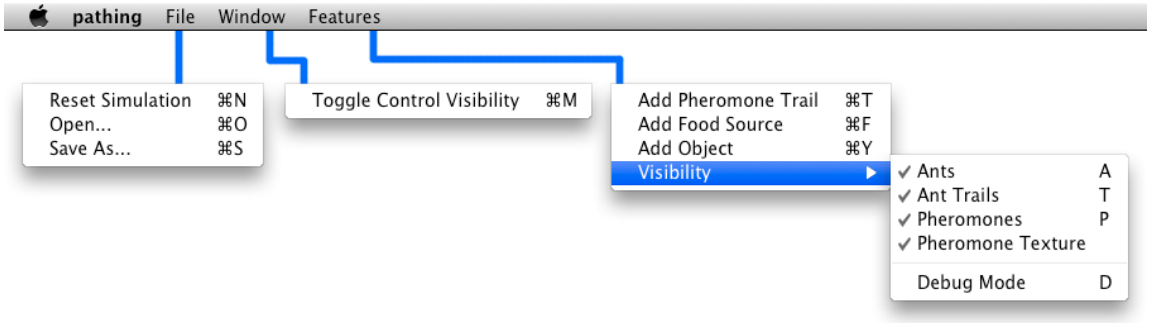

Figure 10: *The menu bar, its submenus, and each action's corresponding key command. (On Linux and Windows systems, the control key is the modifier instead of the command key).*

Stereoscopic Display

On capable systems, the visualization can be displayed stereoscopically. This allows better immersion into the simulated space, as well as an immediate, intuitive understanding of the 3D space in which the ants operate.

Qt's QGL layer determines whether the system is stereo-capable. If the system is capable, the program will switch into stereo mode. In this mode, two cameras are used, one for the left eye image, the other for the right eye. These cameras are linked just as a person's two eyes behave, in that they are a set distance apart and focus on a common point. Each camera renders to a different frame buffer image, allowing the graphics drivers and hardware to handle the specifics of displaying two separate points of view. Since each frame must be rendered twice per frame, the update rate may be lower on stereo capable systems.

No 3D mouse (also known as a spaceball) support is implemented. All camera operations perform in the same manner in both stereo and regular (mono) camera modes, with the addition that the mouse action plane is also the plane on which the two cameras focus.

PROBLEMS ENCOUNTERED

As with any complex programming project, unexpected complications and problems arose throughout the development process. Below are the major problems encountered and their solutions.

Pheromone Discretization Problems

Path spatial preference

Ants creating paths heading to the "left" (if one is using the initial camera placement) will create one or two paths to and from food, and stay mostly linear. Ants heading "right", however, create many interweaving paths (Figure 11). While both behaviors should be possible in the simulation, neither behavior should be prioritized based on direction.

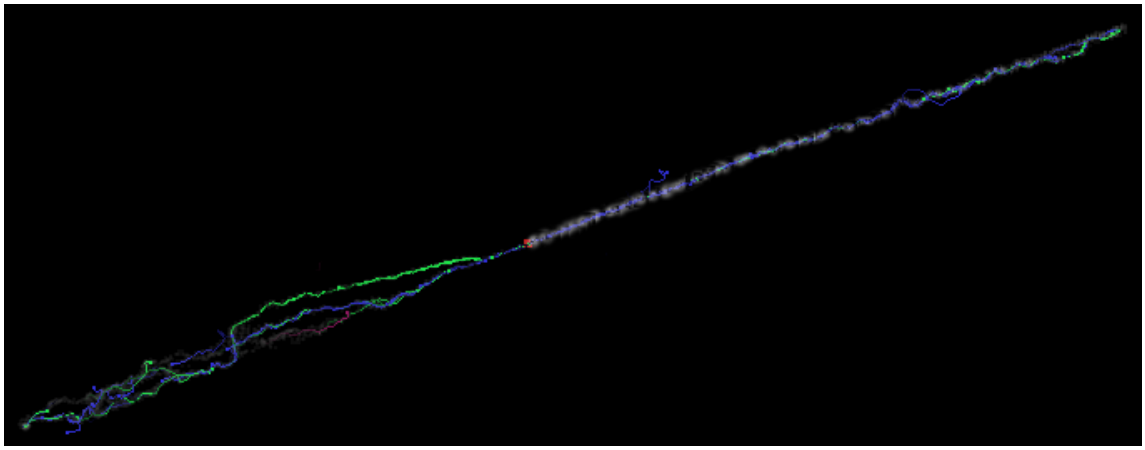

Figure 11: *Path creation behavior was dependent on the direction ants were traveling.*

The pheromone point discretization was originally placing pheromone points slightly below and to the right (in screen space) of the ant, causing ants returning to the nest on the left half of the screen to essentially chase their own paths. This was found to be because the discretization function was acting as a floor function, rounding each spatial location to its lowest set of coordinates. The solution was to add a small constant offset (1/2 the distance between potential spatial locations) to the spatial coordinates when the

discretized pheromone points were converted from their storage location back into 3D space coordinates. This offset moves the pheromone point to the center of its storage location instead of putting it at a corner.

Path holes

Late in the simulation's development, pheromone points suddenly became much more sparse in the scene. Large gaps were visible in the paths, but the ants were still behaving as if the points existed (Figure 12). The problem was diagnosed as a display-related issue, but was difficult to isolate.

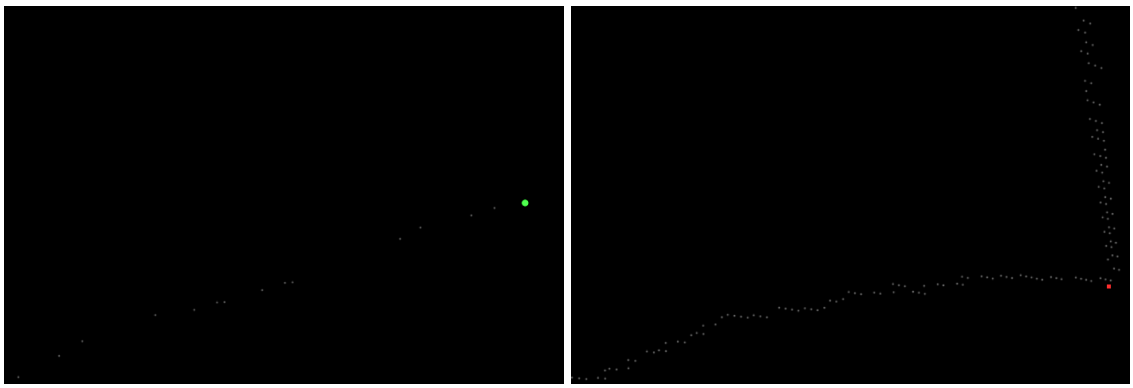

Figure 12: *Pheromone points are sparse (left) before fixing the problem (right).*

The simulation's visualization engine must sort the pheromone points by distance from the camera. The initial method used the Standard Template Library's (STL) *map* type to quickly sort each octree level by distance. The *map* type requires that each data point have a "key" data value and one or more data values associated with the "key". While not explicitly mentioned in the STL's documentation, the *map* type requires unique identifiers and doesn't allow the use of the "double" data type as its "key" value. The "key" value for the pheromones (it's distance to the camera) was converted into an integer, creating many potential duplicate values, which were therefore not being added to the sorting list. Once this limitation was discovered, the sorting algorithm was changed to use the STL type *multimap*, which operates similarly to the STL's *map* type

but allows for doubles to be used, and allows for multiple elements to have the same key.

Path snapping

Ants would only create paths in straight lines at multiples of 45-degree angles – due West, Northwest, North, etc., as in Figure 13. Ants not making or following paths operated freely in space. The error was eventually found to be that when the ant's previous step was discretized, a pheromone point was occasionally placed ahead of the ant's end position. That point would then be followed in the current time step, increasing the likelihood the ant would continue in the same line. This linearity would "snap" in 45-degree increments, so that if the nest traversal was more direct by traversing a different 45-degree orientation, the ant would switch to that orientation. The obvious solution was to prevent the discretization of the line from creating a pheromone point ahead of the ant's end position for each time step.

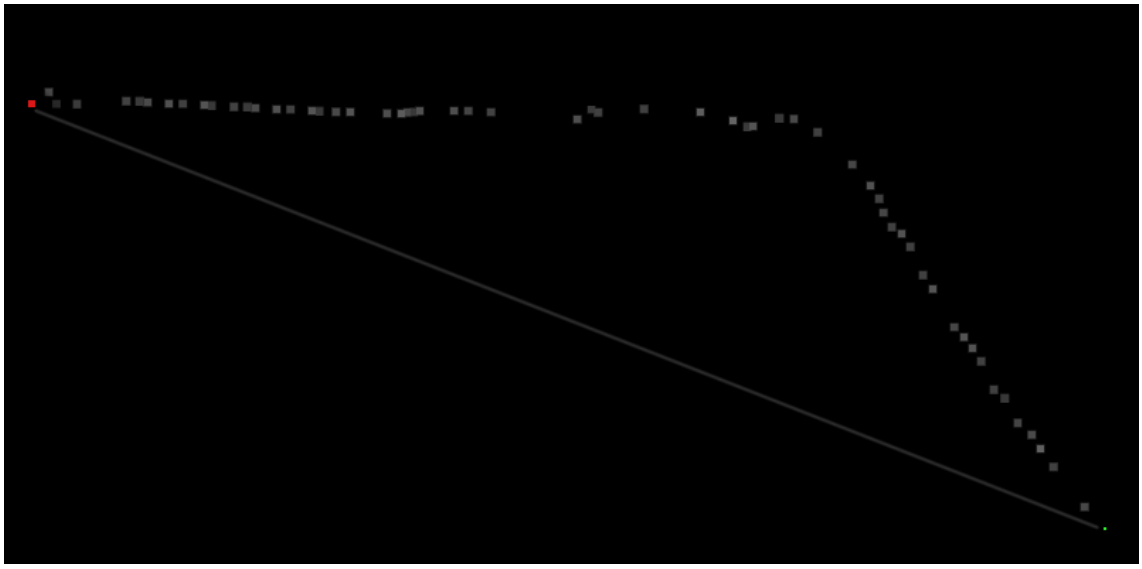

Figure 13: *The pheromone path from nest to food source with the "path snapping" problem unresolved, with a solid line showing the optimal path, which is taken once this problem was resolved.*

Path Backblow

Ants with food would occasionally overshoot the nest a little, laying a bit of pheromone down as they did. Over time, this alternate path would become more attractive than returning to the nest, so all ants would follow this new path, leading to the behavior shown in Figure 14.

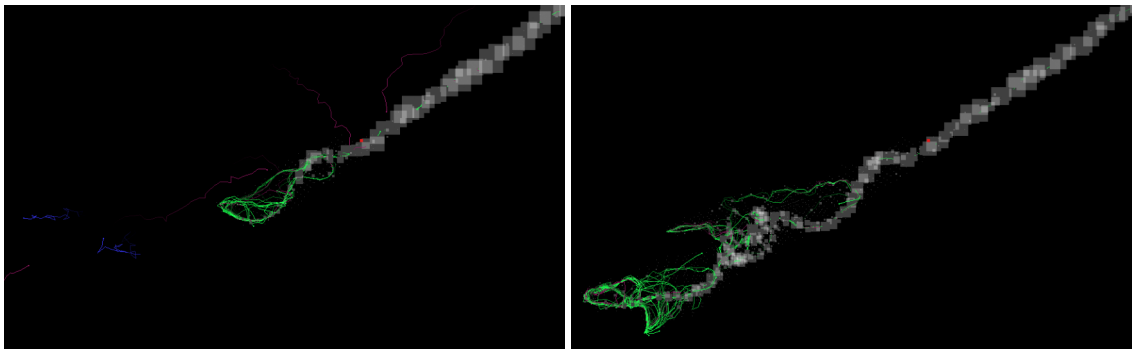

Figure 14: *The path (grey squares) blows back and grows past the nest (red square).*

The solution involved two parts. First, a "ceiling" value was implemented for each pheromone value, meaning that ants could not lay down pheromones at a level above this value. This has a basis in reality since too much pheromone in an area will indicate panic [Grah et al. 2005], a state we are not attempting to invoke in this simulation. The second part of the solution was to increase the ant's attraction to their nest as they get closer to it. Previously, the ants were attracted by a constant amount, which, while simple, was producing unrealistic behavior. This increased attraction meant that once the ants get near enough to their nest to "see" it, they prefer going to their nest rather than following the pheromone paths.

Object Collisions

Edges

When object collisions were first introduced to the simulation, ants following a pheromone path that traversed the object's surface would not successfully cross sharp polygon boundaries, instead retreating back along the pheromone path in the opposite direction (Figure 15).

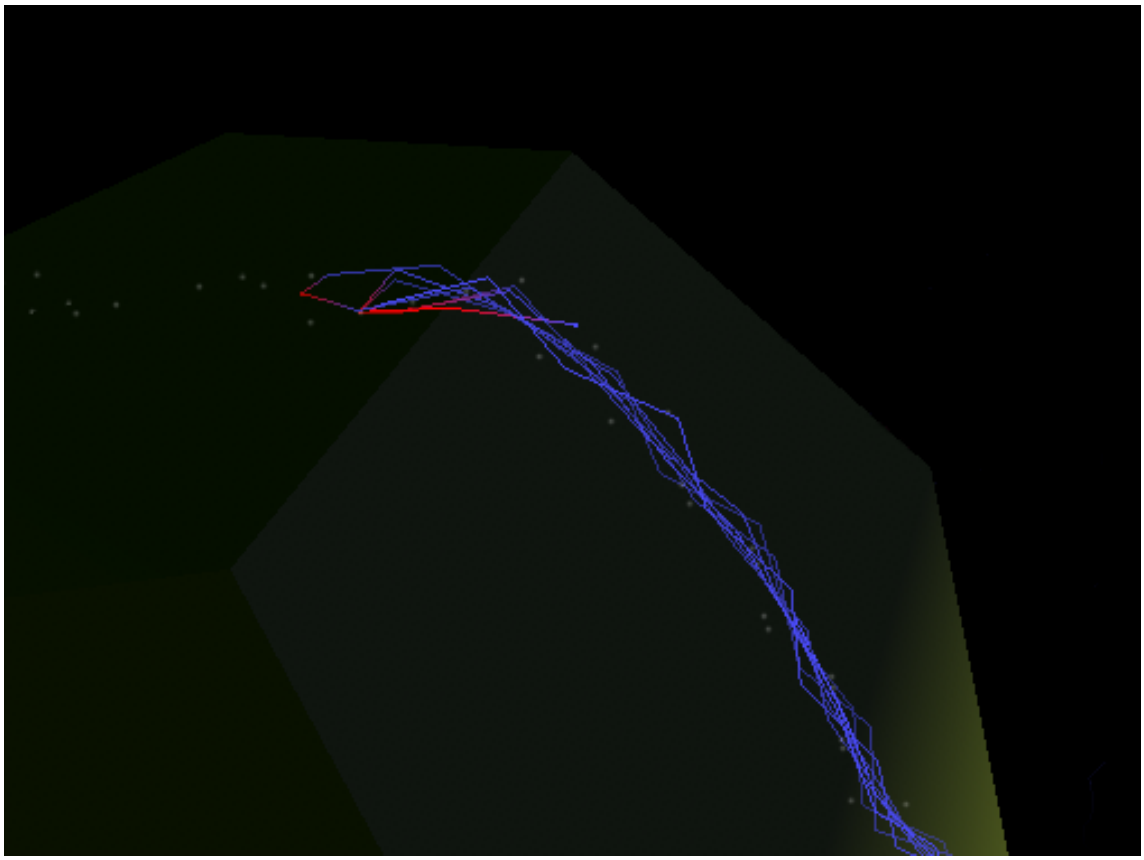

Figure 15: *Ants were unable to cross sharp polygonal edges.*

The ants were emitting pheromones along their path, which could lead to pheromone points being created inside the object when the ant's movement was discretized. While this behavior would occasionally cause problems on polygonal faces (see below), the main problem was when trying to cross polygonal edges. Ants would almost never successfully navigate a polygonal edge. The solution was to detect if a pheromone point was being created on the other side of a polygon. To do this, a line segment is constructed between the current position and the potential pheromone point placement. If the line segment intersects a polygon, the potential pheromone point is inside the polygonal object, and therefore invalid. The pheromone position is rotated 180 degrees using the ant's current position as the origin before being placed in the scene. This has the twofold benefit of keeping pheromones outside of objects and moving the

pheromone path slightly off the object's surface, preventing the ant from grazing a polygonal edge.

Faces

A related problem is that ants traveling along a pheromone path would collide with a polygonal object's face. The ants would then reverse their path, doubling back along the pheromone path (Figure 16).

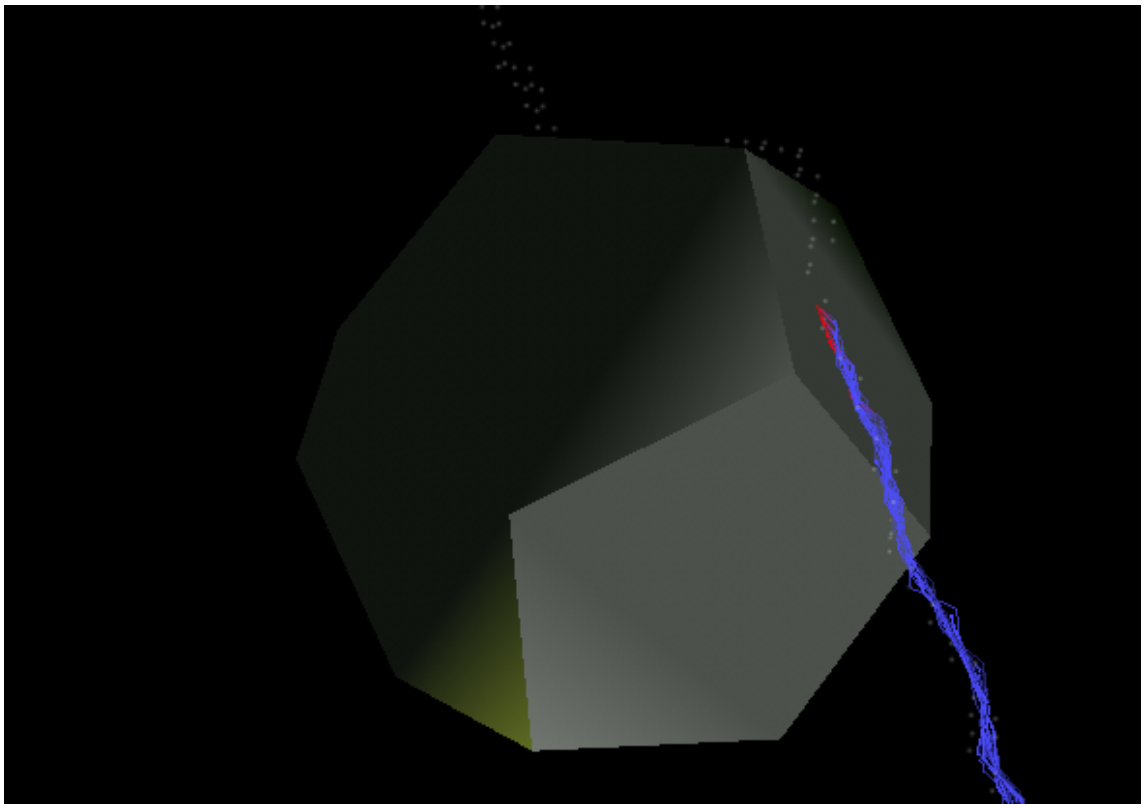

Figure 16: *Ants would collide with a polygon face and turn around.*

This problem required a two-part solution. Creating pheromone points inside of objects made the problem more prevalent, but fixing that problem (as described in the Edges subsection) did not solve the Faces problem, only made it less likely to occur. When an ant hit a polygonal face while following a pheromone path, it still reversed course. The problem turned out to be that the computed vector for bouncing off the polygon used the

nest's position, not the vector towards it, as a component. Fixing this error caused the aberrant behavior to disappear.

RESULTS AND CONCLUSIONS

Simulating ants' behavior would seem straightforward, as there are relatively few rules they have to follow, especially in a simplified world such as this simulation. While the broad strokes of the simulation were fairly straightforward to implement, tweaking the priorities of the ants (for example, how aggressively they followed paths vs. trying to get back to their nest via dead reckoning) to best simulate the desired behavior took many iterations to refine. While numbers extracted from reality would have been preferable, quantifying such concepts from observed data is beyond the scope of this project. Also, the behavior exhibited by the simulated ants seems to closely imitate real-life ant behavior, and is therefore considered qualitatively accurate.

In unrestricted 3D space, which this simulation provides by default, the ants have little trouble returning to the nest after finding their food. The most interesting pathing occurs when multiple obstacles obstruct the direct path (see Figure 17).

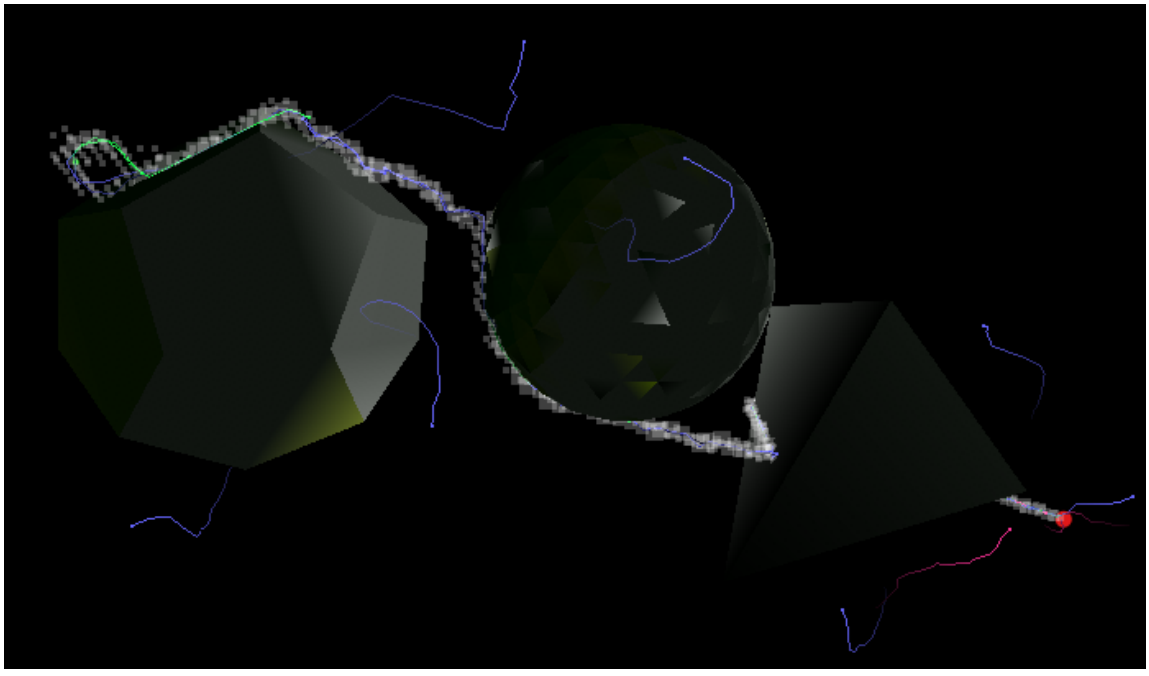

Figure 17: *More objects obstructing the ants' path make for more interesting pathfinding.*

The pseudo-random nature of the simulated ants' movements virtually guarantees that no two simulation runs will be identical. Once an ant has found a food source, however, the simulation runs in a fairly predictable pattern. Ants will be more likely to follow the path the first ant created as the path gets stronger.

The initial randomness of the simulation makes reliably demonstrating some behaviors difficult. This may limit the effectiveness of the program as a teaching tool. Fortunately, the ability to save, reload and edit simulation states allows greater demonstrative control. The user can save a state that nearly fits the desired state, increasing the chance that the desired state will happen soon, or save a state that already exhibits a desired behavior (ants moving along a pheromone path, for example). The user could even manually edit the saved state to display exactly the desired state once the saved state file is loaded.

The ants' unpredictable nature can also slow down the debugging process, as it may take many runs to recreate a problematic behavior. Using saved states likely to cause the behavior decreases debugging time.

Ants' behavior, as mentioned in the Implementation section, is achieved by tuning the three primary ant "options" to create different behaviors for searching and returning with or without food. These behaviors were developed through iterative tuning. A slight change in one of the options can create a significant change in the resulting behavior. As such, the option tuners are not made available for users' modification.

When placing an obstacle in the way of an established path, the ants immediately adjust their path around the object. If the obstacle suddenly appears in space (instead of being moved into place) ants occupy, the ants will become trapped inside the object until their life runs out and a new ant leaves the nest. When an object obstructing a path is moved out of the way of the path, the generation of a shorter path is not guaranteed to occur. If the existing path is sufficiently strong, ants will not deviate from it, so the existing path will stay in place.

Stereoscopic visualization was found to be more important to the project than first anticipated. While the ants operate in a 3D environment, screen space for most computers is still essentially 2D. Consequently, the ants seem to operate on the viewing plane instead of the 3D space they are actually in. When viewing the scene stereoscopically, the actual 3D operational space becomes instantly obvious. This provides a more compelling, easily understandable experience than is provided on a traditional monitor.

An octree was a sub-optimal choice of data structure for the pheromone points. The pheromone point sorting algorithm takes the majority of the processing time for the display generation. While caching the sorted pheromone points alleviated this somewhat, a better solution might be to use a data structure better suited to both storing data and displaying it from arbitrary viewpoints in space, such as a KD-tree or a Binary Space Partition tree.

Most programming errors encountered during development stemmed from the discretization of pheromone points, specifically in the conversion from spatial coordinates to octree storage locations. More thorough testing of the pheromone point class, especially discretization edge cases, at early development stages would likely have decreased the development time used dealing with these problems.

Developing the application using the Qt framework made cross-platform programming easier than trying to adapt the program for multiple operating systems independently. Qt handles most of the system-specific programming calls, allowing a programmer to concentrate on the more interesting and unique aspects of the program. Furthermore, Qt has robust support for OpenGL, so that most system-specific graphics calls are handled by Qt as well. These factors lead to less time being spent on lower-level system-specific programming and more time with the actual simulation.

FUTURE WORK

This project uses a single algorithm to handle all aspects of the simulation. Work could be done to create several different simulation algorithms and allow easy swapping between them. This would allow direct comparison of several simulation algorithms, instead of just the differences obtained by adjusting variables of a single algorithm.

Currently, ants are only guaranteed to successfully traverse convex obstacles. Work could be done to improve obstacle avoidance logic to expand the range of valid obstacles. Additionally, ants could be restricted to movements along a surface based on a terrain file.

This simulation could also be used as a test-bed for robots using pheromone-based behavior for exploration. The robots' and pheromone paths' behavior could be tested in this simulation before being programmed into the robots themselves, allowing quick initial simulations and culling of less useful operating parameters.

Pheromone points could be stored in a manner more efficient for display than the current octree structure, greatly speeding up render times.

Work could be done to make the algorithm a more realistic simulation of ant behavior. This might result in a more effective educational tool for the behavior of real ants rather than a simulation loosely based on ants. Some interesting additions could be:

- Detection of other nests and the addition of ant nest interaction, leading to potential nest skirmishes
- The addition of poisonous foods, with different methods of killing (instant, time-delay, trophallaxis delay)
- Addition of environment visualization techniques such as sun position
- Simulation of nest dynamics such as birthing rates

Alternately, work could be done to further abstract or repurpose this simulation, perhaps adapting it to other NP-complete problems and visualizing results in real-time. For example, instructing ants to touch all food sources before returning to the nest would create a simulation of the Traveling Salesman Problem. This could in turn be used as a basis for both packet routing and Job Shop Scheduling.

To better facilitate further work in this area, a compressed file containing all files related to this code has been submitted as a supplement to this document. Please refer to supplement *pathing.zip* for code and executable files.

REFERENCES

Beckers, R., Deneubourg, J.L., and Goss S. 1992. Trails and U-turns in the selection of the shortest path by the ant Lasius niger. *Journal of Theoretical Biology*, 159, 397-415.

Beckers, R., Holland, O. E., and Deneubourg, J. L. 1994. From local actions to global tasks Stigmergy and collective robotics. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, MIT Press, Cambridge, MA, pp. 181-189.

Di Caro, G., and Dorgio, M. 1998. AntNet: distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence*, 9, 317–365.

Dorigoa, M., and Gambardellab, L. M. 1997. Ant colonies for the traveling salesman problem. *BioSystems*, 43, 73–81.

Gould, W. 1747. *An Account of English Ants.* A. Millar, London.

Grah, G., Wehner, R., and Ronacher, B. 2005. Path integration in a three-dimensional maze: ground distance estimation keeps desert ants Cataglyphis fortis on course. *The Journal of Experimental Biology*, 208, 4005-4011.

Hölldobler, B., Wilson, E. O. 1990. *The Ants*. Belknap (Harvard University Press), Cambridge, MA.

MacLennan, B. *Simulation of Resnick Ants*. 2008. Available from http://www.cs.utk.edu/~mclennan/Classes/420/NetLogo3.1.5/Ants.html, (University of Tennessee at Knoxville, TN); accessed 15 May 2009.

Miller, R. (Producer), and Lisberger, S. (Director). *TRON* (Motion Picture). USA: Walt Disney Studios Motion Pictures, 1982.

Nakamura, T. 2006. *The Soprants: Conceptual and technical framework for a 3D interactive game*. Master's Thesis, Texas A&M University.

Nokia. *Qt Whitepaper*. Available from http://www.qtsoftware.com/files/pdf/qt-4.4-whitepaper (Redwood City, CA, 2008), accessed March 2 2009.

Resnick, M. 1994. *Turtles, Termites and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, Cambridge, MA.

Wehner, R. 2003. Desert ant navigation: how miniature brains solve complex tasks. *Journal of Comparative Physiology A*, 189, 579–588.

Wilensky, U. *NetLogo Ants model*. Available from http://ccl.northwestern.edu/netlogo/models/Ants (Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1998); accessed 12 July 2009.

Wright, W., and McCormick, J. 1991. *SimAnt*. Maxis Software, Emeryville, CA.

Zhang, J., Hu, X., Tan, X., Zhong, J.H., and Huang, Q. 2006. Implementation of an ant colony optimization technique for job shop scheduling problem. *Transactions of the Institute of Measurement and Control*, 28, 93-108.

VITA

Benjamin Sutherland

BenSuth@viz.tamu.edu

Visualization Laboratory
C108 Langford Architecture Center
3137 TAMU
College Station, TX 77843-3137

Education

M.S. in Visualization Sciences	Texas A&M University, December 2009
B.S. in Computer Science	Texas A&M University, August 2006

Employment

Graduate Assistant	Texas A&M Immersive Visualization Center August 2006 – December 2009