CMP-7009A Advanced Programming Concepts and Techniques

Project Report - 16 January 2020

Simulating the activities of an Ant colony

Group members:
Alvin Lu

School of Computing Sciences, University of East Anglia

Version 3.1

Abstract

Ants have always been a popular animal that was researched for it's behaviour. Findings from it has benefited areas such as machine learning, algorithm discovery and server optimisation. This project aims to create a realistic ant simulator that can be used be researchers and educators to discover potential behaviours or run simple simulations for demonstrations and research. The result was an Ant simulator that provides customisation of various environmental and ant values that can be controlled at the users pace. Variables such as birth rate and death rate can be changed before the start of a simulation. It uses an intelligent ant path-finding algorithm that utilises surrounding pheromone data, distance from hive and individual memory. The system is replicable and flexible, which could potentially encourage reuse and expansion in the future.

Chapter 1

Introduction

The biological world has always been a source of inspiration for computer models and algorithms. Within successful model developed, swarm intelligence have always been a topic of interest spanning across wide range of disciplines [1]. Swarming are commonly associated with ants that exhibits large amount of behaviours that can potentially be used to optimise algorithms. Current implementations has managed to replicate certain behaviours of swarming in ants in technologies [2] [3] and machine learning [4]. To provide visualisation and convenience, simulators [5] [6] [7] that mimics the certain behaviour of ants are being developed to aid in the research of ants. Although these implementations have successfully replicated a single case or a portion of ant behaviour, there aren't software available that simulate ants realistically in the biological world.

To tackle the gap, this project aims to develop a realistic ant simulator that allows customisation of various biological values to allow simulation of different ant colonies in varying environments. It will provide a method to show an accurate representation of ant behaviours, simulate survivability, behaviour and interaction of ants in different settings. The solution aims to provide biological and software researchers a tool to visualise different ant colonies and reveal emergent properties that may benefit future swarm intelligence.

1.1 Report structure

This report will provide a description of the design and implementation considerations of the solution. Chapter 1 will provide an introduction to the topic which will be followed by chapter 2 that provides scientific background, previous materials and motivation of for the project. Chapter 3 will detail the design considerations for the solution, supported by analysis of previous literature and existing solutions. Implementation details will be explained in chapter 4 showing figures of the solution and design modifications made. Chapter 5 will provide testing and evaluation of the final software with discussions on it's strengths and weakness will be provided in chapter 6. The report will the be summarised within chapter 7 providing a conclusion to the project.

Chapter 2

Background

2.1 Background information

Swarming is a behaviour that is commonly found in social animals [8]. The concept of swarming is having a large group of individually operating unit (agents or individual animal) working together to achieve a task that is typically too great for a singular unit to complete on its own [9]. There are large amount of research conducted to explain the evolutionary benefits of such behaviour and researchers are attempting and successfully to adapted some of these properties into various computer systems. This is commonly achieved by having a group relatively simpler agents that can communicate between each other instead of a singular system to complete tasks collectively by converging individual accomplishments [10].

Within animals that exhibit swarming, ants are one of the insect that has been vigorously studied for their behaviour. Ants most commonly utilise swarming during the process of foraging. Certain species of ants are shown to leave pheromones to share information an individual ant has learnt [11]. The two most common pheromones are home pheromones that can guide them home and also pheromones that guide other ants to food sources [11]. The pheromones can vary in their duration and potency, which allows transfer of information such as recency and importance of the information [11]. Ants are also shown to retain memories of places it has seen, using visual cues to guide them to their destination [12] [13] [14]. Within certain species of ants, they are shown to have extra ways to find their ways such as using polarized light [15], magnetic [16] [17] and sun [18] compass.

2.2 Existing methods

According to Liviu [5],here were several simulators that were developed in the past. Liviu have also developed an ant simulator that is available online publicly that uses the MASON simulation toolkit [19]. The simulator replicates ants movements using pheromones alongside with the list of features below:

- Allows customisation of certain values such as the ant population and evaporation rate.
- Allows pausing, playing and speeding up or slowing down the simulation.
- Shows the pheromone and ant data of a particular point in the map when clicked

The simulator provides quick and easy way to visualise movements of the ants however it the customisation was limited as the location of the hive, food and obstacle type can not be changed. This simulator aims only to show how ants eventually converge into a path from hive to food shown in figure 7.2 which limits the flexibility of the simulator. The source code of the ant simulator is available on *GitHub* for reference.

Chapter 3

Design

3.1 MoSCoW

3.1.1 Must

Requirements	Description
Independent ant movements	Ants must move independently from each other
Creating hive	Allow creation of a hive. The hive will be the starting point of all ants belonging to that hive.
Static food	Allow creation of static food items on the map.
Pheromone types	Contain a pheromone system that allows deposition of pheromone values on the ground.
Pheromone evaporation	Deposited pheromones should decrease in concentration overtime in a fixed rate.
Obstacle creation	Able to create obstacles that ants can't move on or under
Ant births	Able to dynamically generate ants in with a given birth rate
Ant death	Ants will die and be removed from the map by probabilistic death rate
Pause and play simulation	Allow users to pause the running simulation and replay it multiple times
Shows ant data dynamically	Shows ant population and time elapsed on the UI dynamically

3.1.2 Should

Requirements	Description
Customisable hive location	The hive location of ants should be customisable on allowed locations on the map.
Customisable food location	Food items should be customisable on allowed locations on the map.
Customisable global evaporation rate	Rate of evaporation of pheromones can be customised
Reset simulation	Allows resetting the simulation with different values
Time scaling	Automatically scales the time to reflect realistic values. Allow users to enter actual values for attributes that are connected to time.

3.1.3 Could

Requirements	Description
Multiple hives	Allows creation of multiple ant colonies with varying population
Customizable terrain	Terrains can be customised by the users by painting pixels on the map
Moving food item	Allows creation of food item that moves across the screen
Predator	Allows creation of predators that attacks and consumes the ants
Customisable individual evaporation rate	Allows customisation of individual evaporation rates
Preset ant settings	Provides a list of settings that reflects actual values of ant species
Varying ant types	Allows creation of different types of ant within a colony
Pheromone data at point	Outputs the pheromone data of a particular point in the map
Creation of new pheromones	Allows creation of pheromones with different properties

3.1.4 Won't

Requirements	Description
Detailed UI models	The UI of different items(sprites) in the simulator will not have detailed models and will be represented by simple shapes
In-built screen recording	The system will not provide recording features. As most machines support screen recording internally, implementation of it was not considered in the solution.
Nupital flight simulation	The system will not simulate the process of nupital flight where new queens and male ants form a new hive.

3.2 Structure of the system

The system will have different classes assigned with various responsibilities. The **Display class** will be the creator of the simulator. This class will be called first during execution of the Java executable. This class will create the windows(stages) during instantiation of the simulator. All information of the simulator such as time, all sprites and are saved in the **Ant_Simulator class** shared to other classes when required. After the simulator has been initiated, the **Menu_Controller class** will be assigned the ability to run, modify and terminate the simulator. Inheritance was also taken into consideration during the design. Operating classes within the simulator were developed from a few base classes such as Simulator, Sprites and Ground_Data class. This allows the improves code reusability if other simulators are to be built in the future. The full class diagram can be found in figure 7.12 in the appendix.

The state diagram of individual ants are shown in figure 3.1 and their movement AI is detailed in section 3.3. All ants have a lifespan that is set during initialisation of the simulator. Death of ants are calculated by probability, the closer they are towards the lifespan, the higher the chance of death. Dead ants are removed from the simulator through the handler.

3.3 Ant movement AI

The ants rely on two type of pheromones, home pheromone that is used by ants carrying food to find their way home and food pheromone which is left my ants carrying food and followed by foraging ants to the food source. The pheromones will be the main source

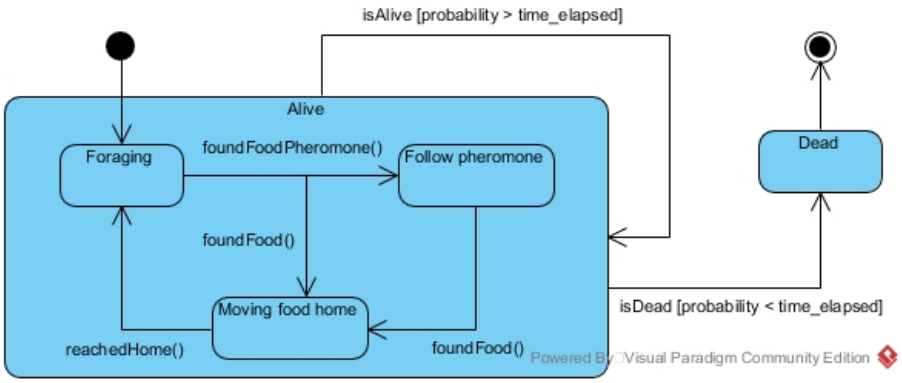

Figure 3.1: State diagram of an Ant

of navigation however, the ants will also rely on the distance between their location and the hive and will try to move closer to the hive if possible, mimicking the process of path integration [20]. Relying on hive distance will ensure that ants that are trying to navigate home will not get stuck following the trails of other foraging ants that has moved further away from the hive.

All ants can move to the 8 potential directions shown in figure 3.2. If there isn't any pheromone data found in their surroundings, the ants will move in random directions but they will prioritise the three direction that the ant is in front.

During initial prototype and testing, It was found that ants in the simulator that relies purely on pheromones for navigation might result in ants being stuck in a cyclic movement if the food source is located further away from the hive where surrounding home pheromones may have evaporated or the density of ants are not high enough to ensure navigation back to the hive. Due to this, individual ants are designed to retain a set amount of locations that it has recently visited and will avoid moving back to the locations unless necessary.

Algorithm 1 shows the high level movement pattern of ants. Algorithm 2, 3 and 4 shows the algorithm used for ants to find their way back to their hive when carrying food. Movement patterns when ants are following food pheromones to food source are similar to following home pheromones back to the hive.

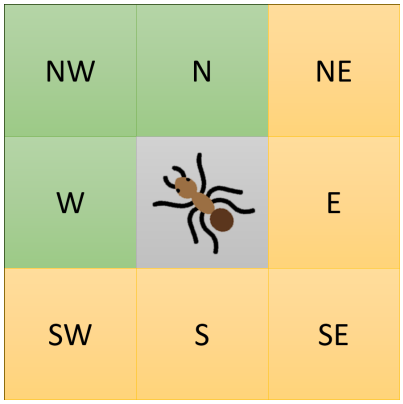

Figure 3.2: Directions an ant can move to if unobstructed. Ants will prioritise moving towards the three directions in front of them (Ant facing NW). Ant image sourced from *Stockio*

Algorithm 1 Ant foraging algorithm

```
if HasFoodItem then
    Find-Direction-Home
else if FoundFoodPheromone then
    Follow-Food-Pheromone
else
    Random-prioritised-movement
end if
```

Algorithm 2 Find-Direction-Home algorithm

```
if Ant reached the hive then
    Drop food
    HasFoodItem ← False
else
    direction ← Decide-Direction-Home-With-Pheromone
    if direction = NULL then
        direction ← Decide-Direction-Home-Without-Pheromone
    end if
end if
moveAnt(direction)
```

Algorithm 3 Decide-Direction-With-Pheromone algorithm

```
potential-directions ← { }
while there's possible directions do
    if direction is not blocked then
        if direction is closer to hive & direction wasn't visited recently then
            if direction-contains-pheromone then
                potential-directions +← direction
            end if
        end if
    end if
end while
if potential-directions = NULL then
    return NULL
else
    direction ← direction-with-maximum-pheromone(potential-directions)
    return direction
end if
```

Algorithm 4 Decide-Direction-Without-Pheromone algorithm

```
potential-directions ← { }
while there's possible directions do
    if direction is not blocked then
        if potential-directions = NULL & direction wasn't visited recently then
            potential-directions +← direction
        else if direction is closer to hive then
            potential-directions +← direction
        end if
    end if
end while
direction ← direction-closest-to-home(potential − directions)
return direction
```

Chapter 4

Implementation

4.1 Platform and packages

The simulator is developed in Java using the IDE *IntelliJ*. Using Java as the development language ensures that the solution can be run on a wide range of Operating Systems while the object-oriented paradigm allows definition of classes that's duplicable and inheritable. This would be beneficial for this project as the simulator involves creation of large amount of objects that are similar structurally but has to behave independently. The UI will also be developed using *JavaFX* as its platform. JavaFX provides the ability to separate UI from the implementation logic using FXML. It allows quick development of simple user interface that can be easily expanded and updated as JavaFX also supports CSS and HTML.

4.2 Development stages

The solution will be developed from scratch in development stages. The decision to do this was because the previous simulator was developed using MASON toolkit which causes the software to have dependency on the toolkit to operate. This created limitations to the portability of the system. The details of different stages is detailed within this section and screen captures of different versions are located in the appendix.

4.2.1 Stage 1

Stage 1 focuses on creation ants that can move and stay within the boundaries and moves randomly with prioritisation of the three directions in front of them. All values in the simulator are preset. Figure 7.3 in the appendix shows a screen capture of stage 1 development

4.2.2 Stage 2

Stage 2 implements the ants capability of locating food and brining it back home. During this stage, the ants will rely on path integration and attempt to move closer to the hive on each tick. The distance to the hive is calculated by euclidean distance. This stage focuses on the ants capability of state change.

4.2.3 Stage 3

Stage 3 focuses on pheromone deposition. According to the state the ants is in, they will deposit either home or food pheromone and the pheromones will evaporate at a fixed rate over time. Movement of ants was also modified to rely on pheromones for path finding.

4.2.4 Stage 4

Obstacles was implemented during stage 4. Additional obstacle data was added onto the ground data which is has a unique property that stops ants from moving across or over it. To adapt to it, ant movement algorithm was further expanded to include obstacle avoidance.

4.2.5 Stage 5

Stage 5 involves the implementation of controller window, modifiable values and extra attributes such as birth rate and death rate. Ant movement algorithm was further improved to include local memory to improve the efficiency.

4.3 Final product

4.3.1 User interface

The user interface contains two windows, the simulator window and control window (figure 4.2). The simulator window contains all the sprites and will be showing the ant movements when the simulator is initiated. The window responds to mouse clicks and will add the home location and food location during set up and returns the pheromone data at the clicked point when the simulator is running. The control window will provide instructions during setup, prompting the user to select the locations of hive and foods. After the simulation has started, the window provides settings that can be modified and simulator controls. Figure 4.1 and 4.2 shows the user interface.

4.3.2 Features

The simulator will initially be created with default values. Prompting the user to pick a location for the hive and food. After first instantiation of the simulator, the users are allowed to change the settings listed below:

- Individual ant settings - The lifespans, birthrate and number of ants can be modified according to values of different ant species.
- Environmental values - Evaporation rate, number of food source, types of obstacle and the scale of time can be modified to simulate different weathers and conditions of the environment the ant is living in.
- Speeding up the simulation - Whilst the user can implement a fixed time scaling value for the simulator, the speed can be dynamically sped up and slowed down.
- Obtaining pheromone data - Dynamic pheromone data can be obtained by clicking on points on the map.

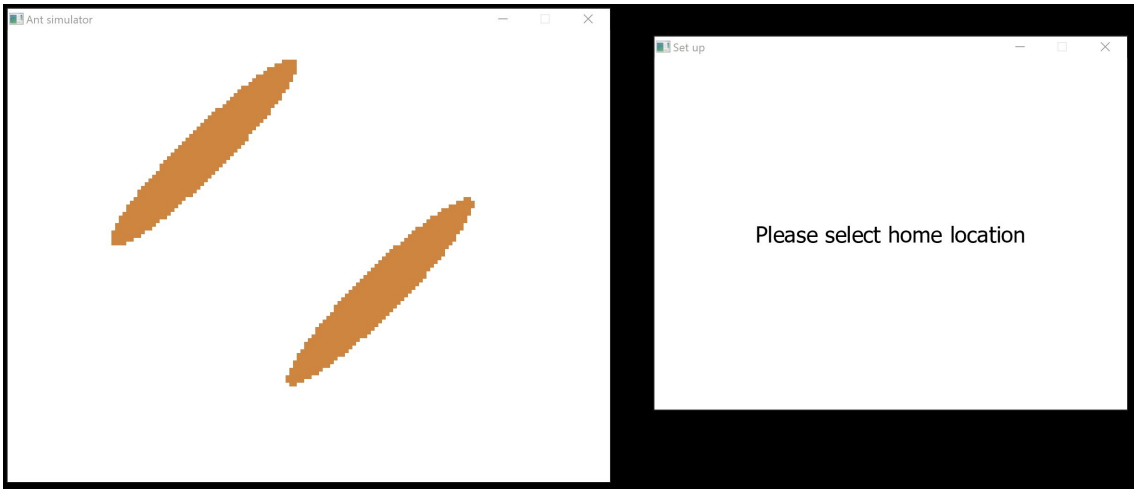

Figure 4.1: Selecting locations for hive and food.

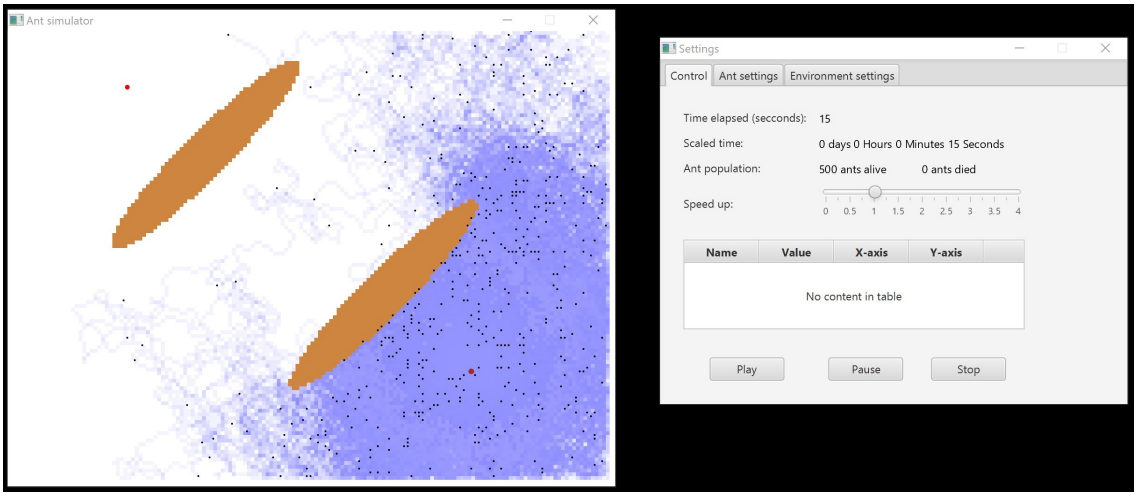

Figure 4.2: Main UI for the ant simulator. The simulator window is located on the left and the control window is on the right.

Chapter 5

Testing

As the design of the static classes are straightforward, testing is conducted heavily towards the end of the development prioritising Dynamic black box and white box testing. Bugs and issue are predicted to be mostly emergent, caused by the ants interaction with its environment.

5.1 Unit testing

For unit testing, values that are passed to and object are checked using console output to print out the values. Values printed out on the console are shown in figure 7.9 and 7.10 in the appendix. A test class was also created to test out new methods and packages before integrating it into the system (Figure 7.11). In these unit tests, expected and unexpected values are entered to the methods to test if the return values are expected.

5.2 Integration testing

Integration testing focuses on parameter passing from the UI to back-end and vice-versa. This is achieved by firstly outputting the values passed from the UI to other classes to the console and checking if the input matches the output. Integration testing was also conducted with each stage of implementation. The simulation was executed a left for a duration to reveal potential bugs with each addition of features.

5.3 User testing

Three users were selected to conduct black box testing on the finished software. The users were initially not given any information on the software except the main aim and was tasked to interact with the software. Most users managed to utilise the simple function of the software (e.g. pausing and playing) however they struggled with modifying the values. Descriptions of the value were then modified to provide better guidance for the types of input a user can put in. The users were then instructed to complete the tasks listed below:

- Speed up the ants movement
- Modify the ants settings
- Increase the time scale of the simulator
- Get the pheromone values of a particular coordinate

All bugs encountered during testing was recorded and fixed after the test.

Chapter 6

Discussion and Evaluation

From a review of the MoSCoW analysis, the ant simulator has managed to fulfil all the requirements in must and should. It also managed to achieve a requirement in the could section such as obtaining pheromone data at a point on the map. The simulator is able to provide simulations with a range of different values. Compared to the MASON Ant simulator [5], this solution provided a more customisable ant simulator with a larger map. This allows a wider variation of simulations within a larger area.

However, during stress testing, the simulator is shown to slow down significantly or freeze when the population of ants goes past 2000. Future improvements could introduce parallel programming for processing of ant movements to allow the simulator to accommodate larger ant populations. During final testing, the ants were able to converge into path between the food source and the hive even with multiple food sources (figure 7.8). The ants do not show cases of cyclic movement but there are cases where ants struggle to find a path home. Ants will trace the sides of obstacle when attempting to locate the path home which consumed time and affected the results of when converging to an optimum path. Future improvements can be made to improve the decision making of the ants during movements to include more factors described in chapter 2 to ensure that optimum paths can be found more efficiently. Additionally, some values in the simulator are currently not customisable such as the potency of pheromones and individual rates, future improvements could consider implementing these as customisable values.

Due to the structure simulator was built, this software provides high flexibility when it comes to scaling and reusing. Similar simulators can potentially be built using this software as it's base, providing opportunities for large amount of future work.

Chapter 7

Conclusion and future work

This ant simulator provided a simple implementation for running a small scale ant colony that behaves similarly to real life ants. It utilises path utilisation, pheromone trails and visual memory to create an intelligent movement pattern for the ants. The simulator also showcased the interaction of different simplistic agents to complete a task together.

For future work, I intent to to improve the ant path-finding algorithm to better match it's biological counterparts and expanding it's features to accommodate a larger range of values, environments and species.

Bibliography

[1] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *From Natural to Artificial Swarm Intelligence*. Oxford University Press, Inc., USA, 1999.

[2] Sabine Hauert, Laurent Winkler, Jean-Christophe Zufferey, and Dario Floreano. Ant-based swarming with positionless micro air vehicles for communication relay. *Swarm Intelligence*, 2(2-4):167–188, 2008.

[3] Eugene Thacker. Networks, swarms, multitudes (part two). *CTheory*, pages 5–18, 2004.

[4] Julia Handl and Bernd Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113, 2007.

[5] Liviu Panait and Sean Luke. Learning ant foraging behaviors. Citeseer.

[6] Liviu Panait and Sean Luke. Ant foraging revisited.

[7] Liviu Panait and Sean Luke. A pheromone-based utility model for collaborative foraging. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pages 36–43. IEEE, 2004.

[8] Veysel Gazi and Kevin M Passino. Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):539–557, 2004.

[9] Olaf Witkowski and Takashi Ikegami. Emergence of swarming behavior: foraging agents evolve collective motion based on signaling. *PloS one*, 11(4):e0152756, 2016.

[10] Gerardo Beni. From swarm intelligence to swarm robotics. In *International Workshop on Swarm Robotics*, pages 1–9. Springer, 2004.

[11] Duncan E Jackson and Francis LW Ratnieks. Communication in ants. *Current biology*, 16(15):R570–R574, 2006.

[12] Rudiger Wehner and F Räber. Visual spatial memory in desert ants, cataglyphis bicolor (hymenoptera: Formicidae). *Experientia*, 35(12):1569–1571, 1979.

[13] Robert A Harris, Natalie Hempel de Ibarra, Paul Graham, and Thomas S Collett. Ant navigation: Priming of visual route memories. *Nature*, 438(7066):302, 2005.

[14] Sophie EF Evison, Owen L Petchey, Andrew P Beckerman, and Francis LW Ratnieks. Combined use of pheromone trails and visual landmarks by the common garden ant lasius niger. *Behavioral Ecology and Sociobiology*, 63(2):261, 2008.

[15] Rudolf Jander and Ursula Jander. The light and magnetic compass of the weaver ant, oecophylla smaragdina (hymenoptera: Formicidae). *Ethology*, 104(9):743–758, 1998.

[16] Eliane Wajnberg, Geraldo Cernicchiaro, and Darci Motta de Souza Esquivel. Antennae: the strongest magnetic part of the migratory ant. *Biometals*, 17(4):467–470, 2004.

[17] Darci MS Esquivel, Daniel Acosta-Avalos, Léa J El-Jaick, Alexandra DM Cunha, Maria G Malheiros, Eliane Wajnberg, and Marilia P Linhares. Evidence for magnetic material in the fire ant solenopsis sp. by electron paramagnetic resonance measurements. *Naturwissenschaften*, 86(1):30–32, 1999.

[18] Martin Müller and Rüdiger Wehner. Wind and sky as compass cues in desert ant navigation. *Naturwissenschaften*, 94(7):589–594, 2007.

[19] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[20] Allen Cheung, Lex Hiby, and Ajay Narendra. Ant navigation: fractional use of the home vector. *PLoS One*, 7(11):e50451, 2012.

Appendix A

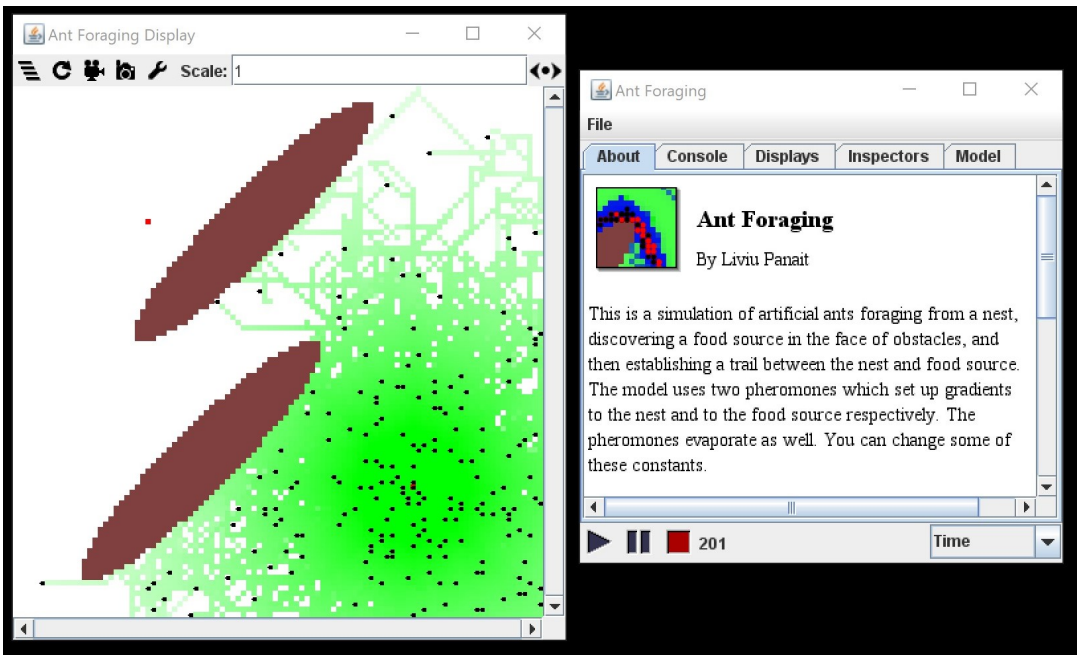

Figure 7.1: Ant simulator developed by Liviu Panait [5]

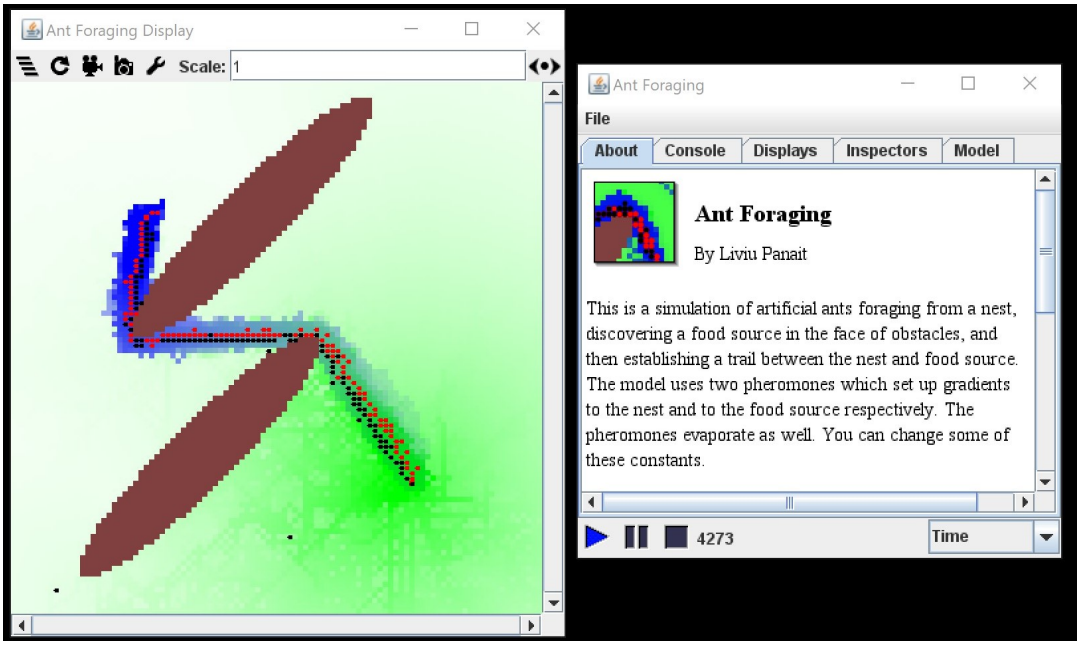

Figure 7.2: Convergence of ant in the simulator [5]

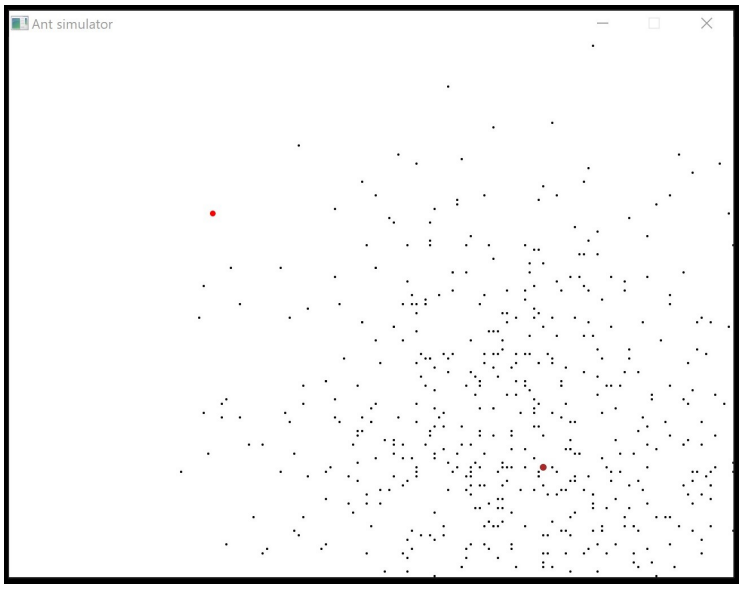

Figure 7.3: Stage 1 of development. The black dots shows the ants and the red dots are the hive and food.

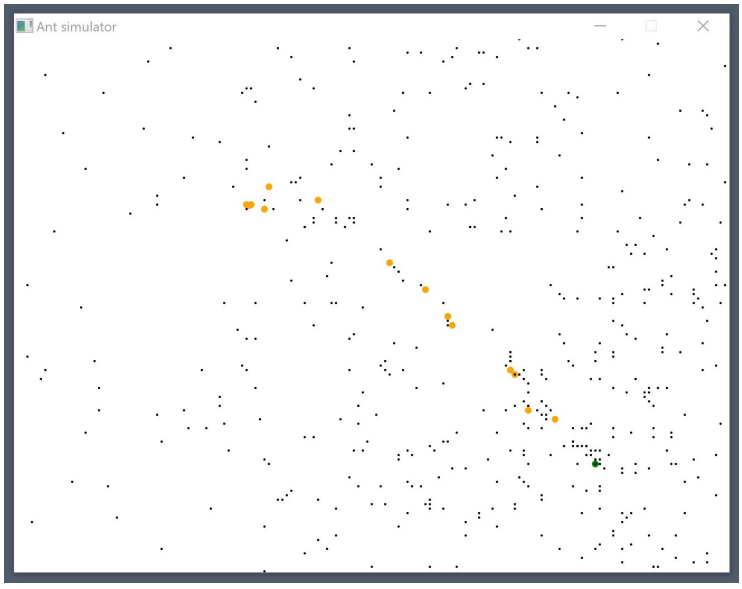

Figure 7.4: Stage 2 of development. Ants are able to locate food and bring it back to the hive.

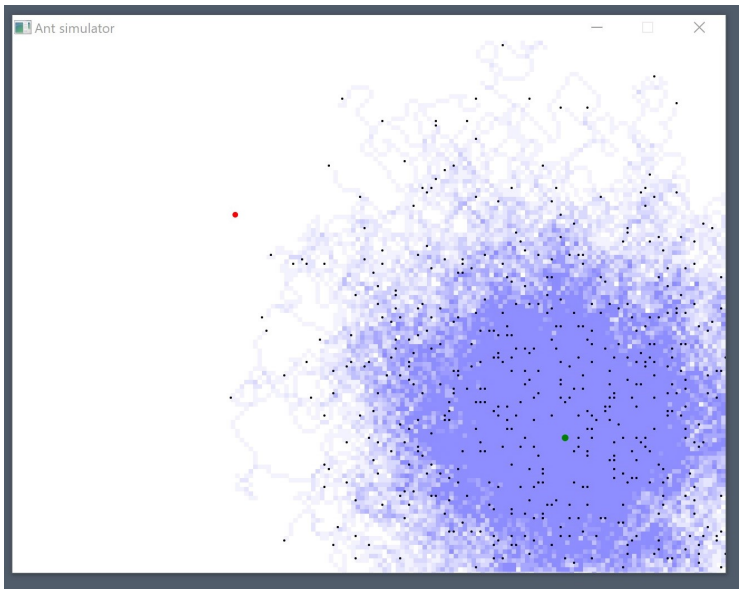

Figure 7.5: Stage 3 of development. Ants leave pheromones while foraging.

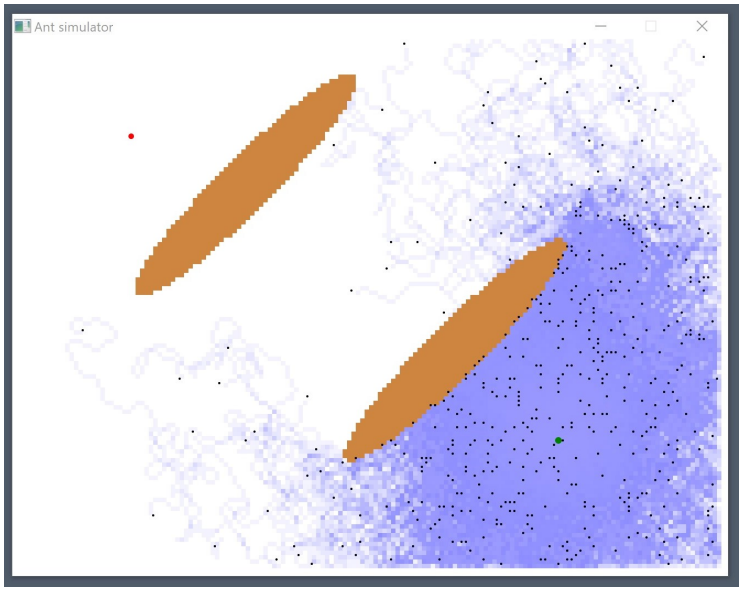

Figure 7.6: Stage 4 of development. Ants movement involves avoiding obstacles.

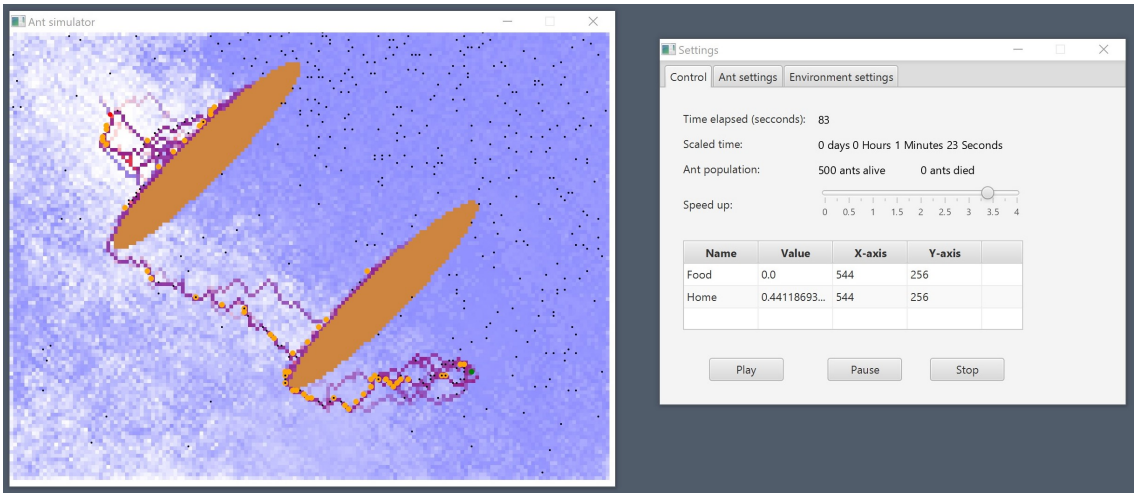

Figure 7.7: Stage 5 of development. Implementation of control UI and improving ant path-finding.

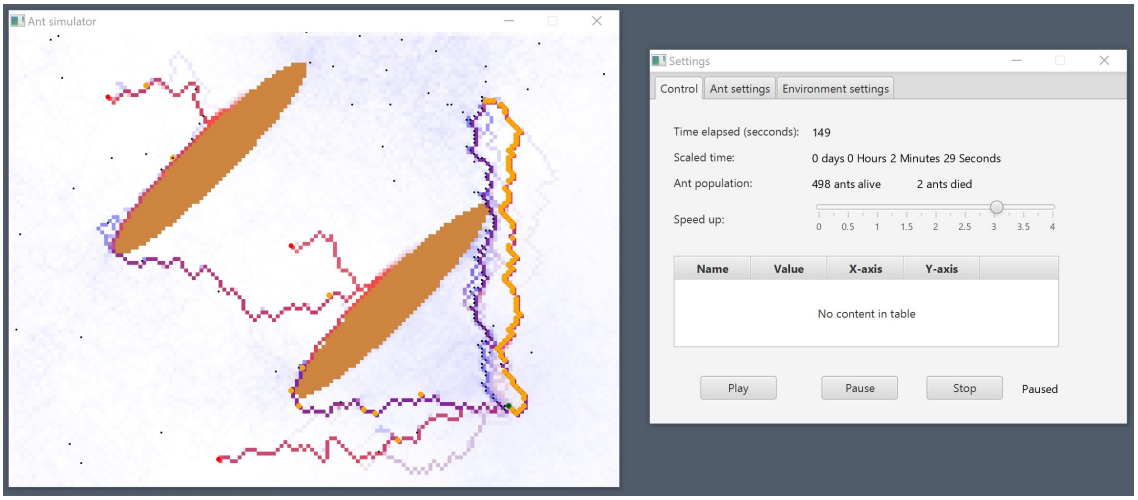

Figure 7.8: Convergence of ants across multiple food sources

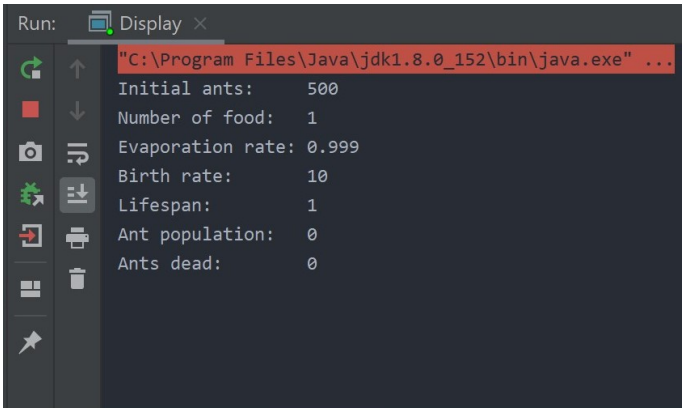

Figure 7.9: Values added into the ant simulator printed out to the console

```
Birth time         :2020-01-15T02:25:41.786Z
Carrying food      :false
Following food     :false
Blocked directions :[8]
Previous direction :NE

Birth time         :2020-01-15T02:25:41.786Z
Carrying food      :false
Following food     :false
Blocked directions :[0, 1]
Previous direction :SW

Birth time         :2020-01-15T02:25:41.786Z
Carrying food      :false
Following food     :false
Blocked directions :[]
Previous direction :W
```

Figure 7.10: Values added into the ants printed out to the console

```
Current time:             2020-01-15T02:31:35.134
Current time + 1 day:     2020-01-16T02:31:35.134
Time difference in days:  1
Time difference in hours: 24
Time difference in mins:  1440
Time difference in milli: 86400000

Process finished with exit code 0
```

Figure 7.11: Using the test class to test the *Instant* and *Duration* class

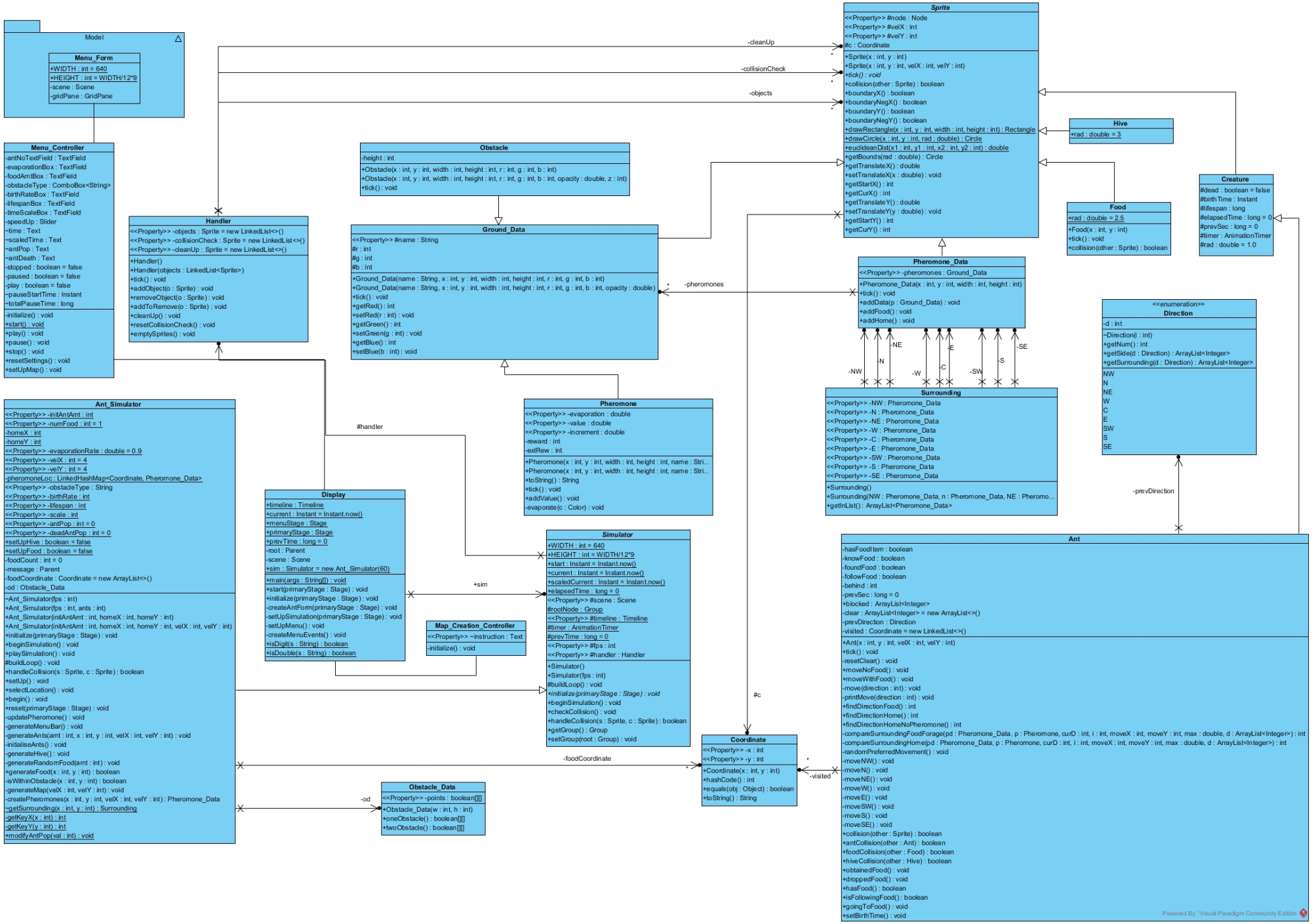

Figure 7.12: Full class diagram of the Ant simulator