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 1.0

Abstract

An abstract is a brief summary (maximum 250 words) of your entire project. It should cover your objectives, your methodology used, how you implemented the methodology for your specific results and what your final results are, your final outcome or deliverable and conclusion. You do not cover literature reviews or background in an abstract nor should you use abbreviations or acronyms. In the remainder of the report the chapter titles are suggestions and can be changed (or you can add more chapters if you wish to do so). This template is designed to help you write a clear report but you are welcome to modify it (at your peril ...). Finally, a guideline in size is approximately 3,500 words (not including abstract, captions and references) but no real limit on figures, tables, etc.

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]. In computing, swarms are usually associated with ants that exhibits large amount of behaviours that can potentially be used to optimise algorithms. Whilst there exist techniques and simulations [2] [3] [4] that mimics the certain behaviour of ants, there aren't software available that simulate ants realistically in the biological world.

This project aims to develop a realistic ant simulator that allows customisation of various biological settings to simulate different ant colonies in varying environments. It will provide a method to show an accurate representation of ant behaviours, simulate survivability or ants in different settings, and reveal emergent properties that may benefit future swarm intelligence.

1.1 Report structure

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 and the report will end with chapter 6 providing a conclusion to the project.

Chapter 2

Background

2.1 Background information

Swarming is a behaviour that is commonly found in social animal species [5]. 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 to great for a unit to complete on its own [6]. Large amount of research has been conducted to explain the evolutionary benefits of such behaviour and researchers are aiming to adapt 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 [7].

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 [8]. The two most common pheromones are home pheromones that can guide them home and also pheromones that guide other ants to food sources [8]. The pheromones can vary in their duration and potency, which allows transfer of information such as recency and importance of the information [8]. Aside from pheromones, ants are shown to retain memories of places it has seen, using visual cues to guide them to their destination [9] [10] [11].

2.2 Existing methods

Currently, an ant simulator was found publicly available online that uses the MASON simulation toolkit [12]. The simulator provides a list of features listed 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 2.2 which limits the flexibility of the simulator. The source code of the ant simulator is available on *GitHub* for reference.

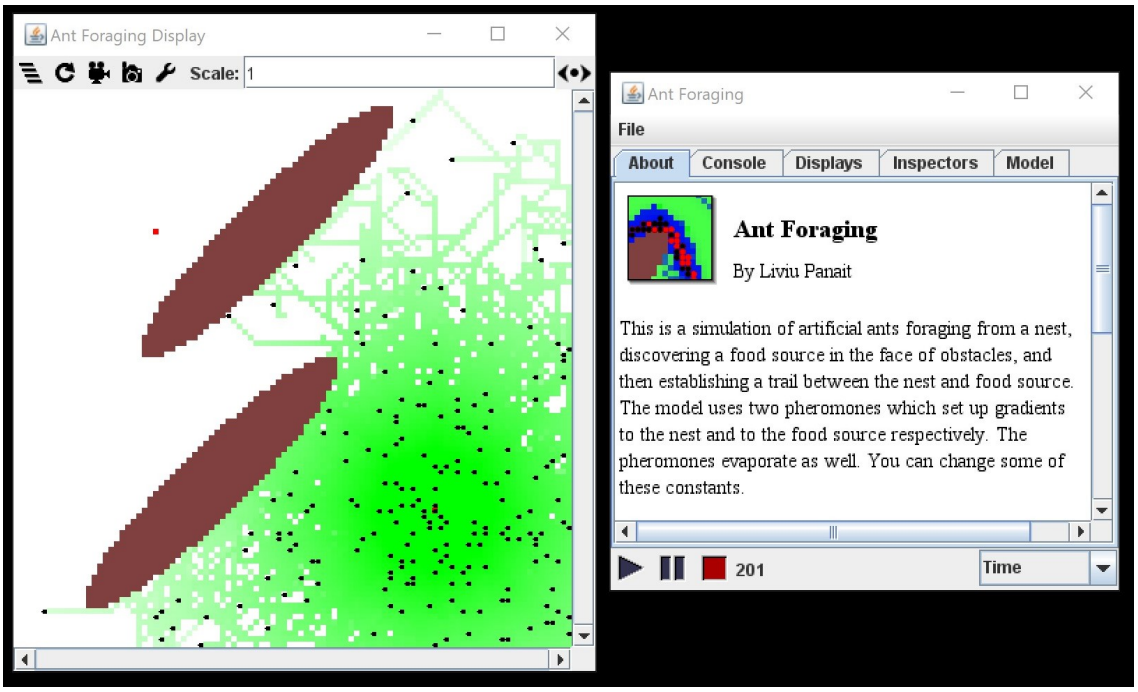

Figure 2.1: Ant simulator developed by Liviu Panait [2]

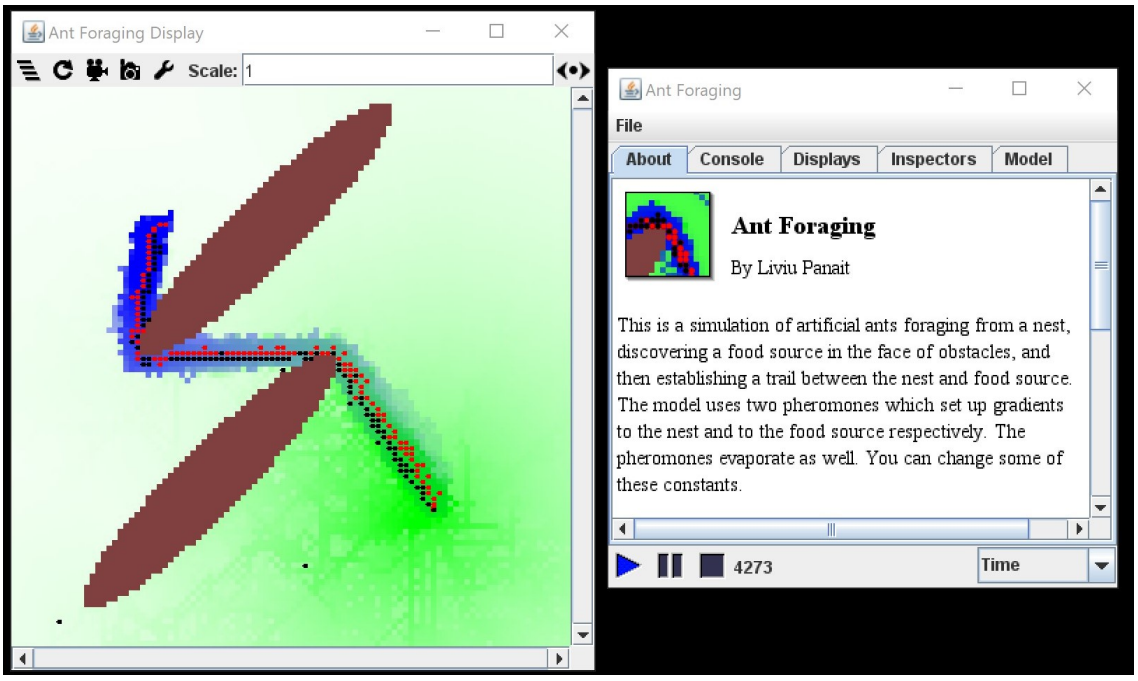

Figure 2.2: Convergence of ant in the simulator [2]

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
Nupital flight simulation	The system will not simulate the process of nupital flight

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) for 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. The system was design with a few base classes that the ant simulator inherits from such as Simulator, Sprites and Ground_Data class. This allows the improves code re-usability if other simulators are to be built in the future. The full class diagram can be found in figure 6.1 in the appendix.

The state diagram of individual ants are shown in figure 3.1. 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

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 analysis of MASON ant simulator and literatures associated, It was found that the the ants in the simulator relies purely on pheromones for navigation which might result in ants being stuck in in a cyclic movement if the food source is located further away from

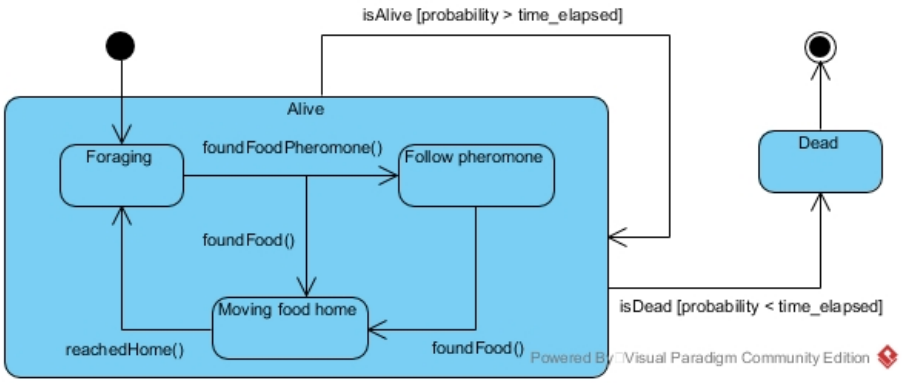

Figure 3.1: State diagram of an Ant

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.

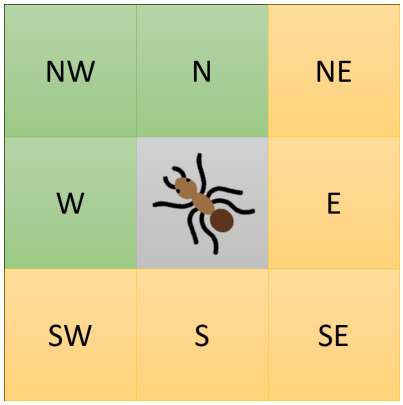

Figure 3.2: Directions an ant can move to if unobstructed. Ants will prioritise moving towards the three directions in front of them (Ant facing NW). Ant image sourced from *Stockio*

Algorithm 1 Ant foraging algorithm

```
1: if HasFoodItem then
2:     Find-Direction-Home
3: else if FoundFoodPheromone then
4:     Follow-Food-Pheromone
5: else
6:     Random-prioritised-movement
7: end if
```

Algorithm 2 Find-Direction-Home algorithm

```
1: if Ant reached the hive then
2:     Drop food
3:     HasFoodItem ← False
4: else
5:     direction ← Decide-Direction-Home-With-Pheromone
6:     if direction = NULL then
7:         direction ← Decide-Direction-Home-Without-Pheromone
8:     end if
9: end if
10: moveAnt(direction)
```

Algorithm 3 Decide-Direction-With-Pheromone algorithm

```
1: potential-directions ← { }
2: while there's possible directions do
3:     if direction is not blocked then
4:         if direction is closer to hive & direction wasn't visited recently then
5:             if direction-contains-pheromone then
6:                 potential-directions +← direction
7:             end if
8:         end if
9:     end if
10: end while
11: if potential-directions = NULL then
12:     return NULL
13: else
14:     direction ← direction-maximum-pheromone(potential − directions)
15:     return direction
16: end if
```

Algorithm 4 Decide-Direction-Without-Pheromone algorithm

```
1: potential-directions ← { }
2: while there's possible directions do
3:     if direction is not blocked then
4:         if potential-directions = NULL & direction wasn't visited recently then
5:             potential-directions +← direction
6:         else if direction is closer to hive then
7:             potential-directions +← direction
8:         end if
9:     end if
10: end while
11: direction ← direction-closest-to-home(potential − directions)
12: 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 User interface

pr

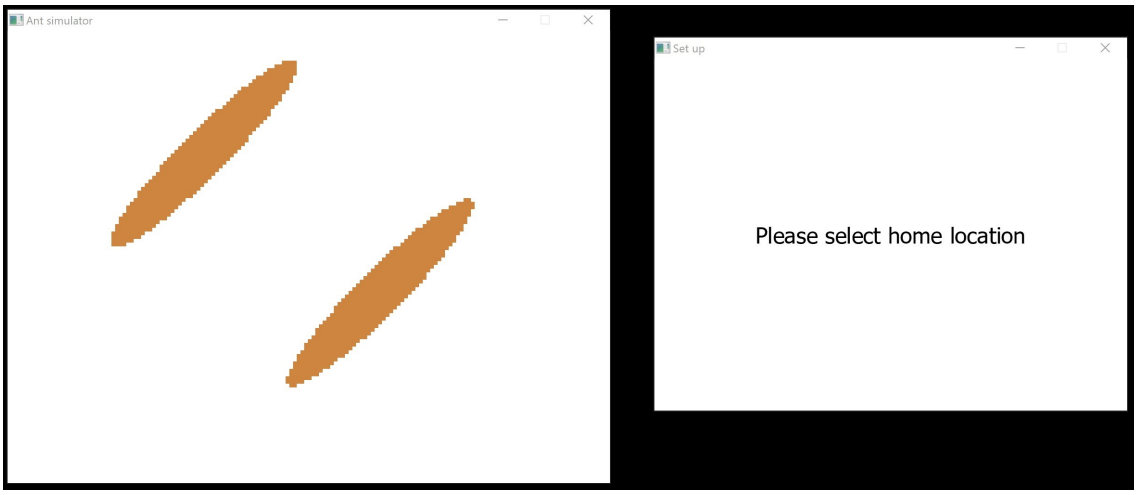

Figure 4.1: Selecting locations for hive and food

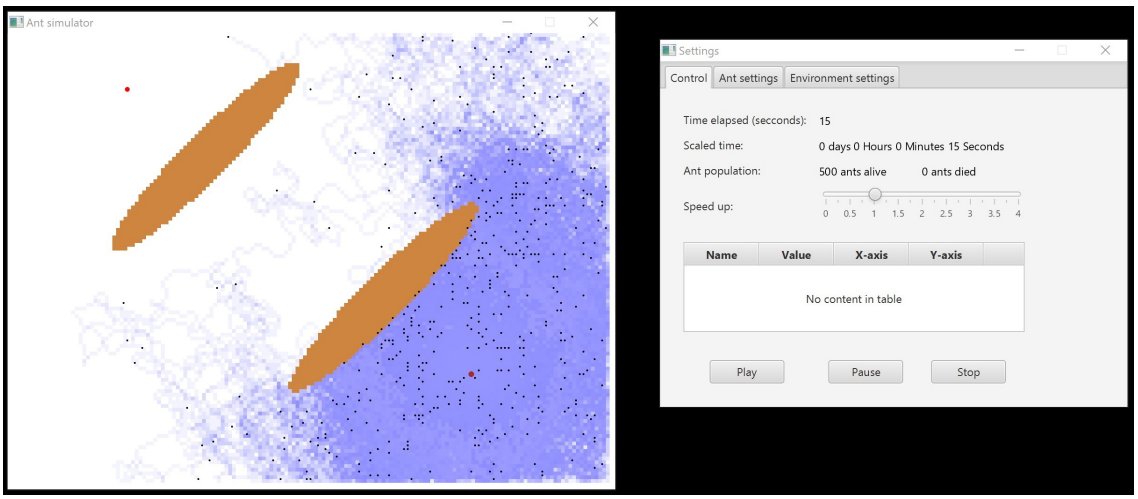

Figure 4.2: Main UI for the ant simulator

Chapter 5

Testing and Discussion

As the software depends heavily on the ants interaction with the environment, testing will be conducted heavily towards the end of the development stage instead of early stages. Bugs and issue are predicted to be mostly emergent, caused by the ants interaction with its environment.

5.1 Unit testing

For unit testing, a test class was created within the package to test advanced method implemented in all classes. Values that are passed to and object are checked using console output to print out the values.

5.2 Integration testing

parameter passing

5.3 User testing

Chapter 6

Conclusion and Future Work

Conclude your achievements and put them in perspective with the MoSCoW analysis you did early on. Be honest by declaring for example S categorised objectives which did not make it to the final deliverable rather than reversely modifying your MoSCoW in Chapter 1! Also discuss future developments and how you see the deliverable improving if more time could be spent. Do not put in subjective opinions or rants or excuses as to why something did not work out as planned in your report. A technical report is not a medium for complaints as there are other outlets for that typically well before you came to this stage (e.g. labs, e-mail etc.).

Bibliography

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

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

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

[4] 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.

[5] 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.

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

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

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

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

[10] 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.

[11] 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.

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

Appendix A

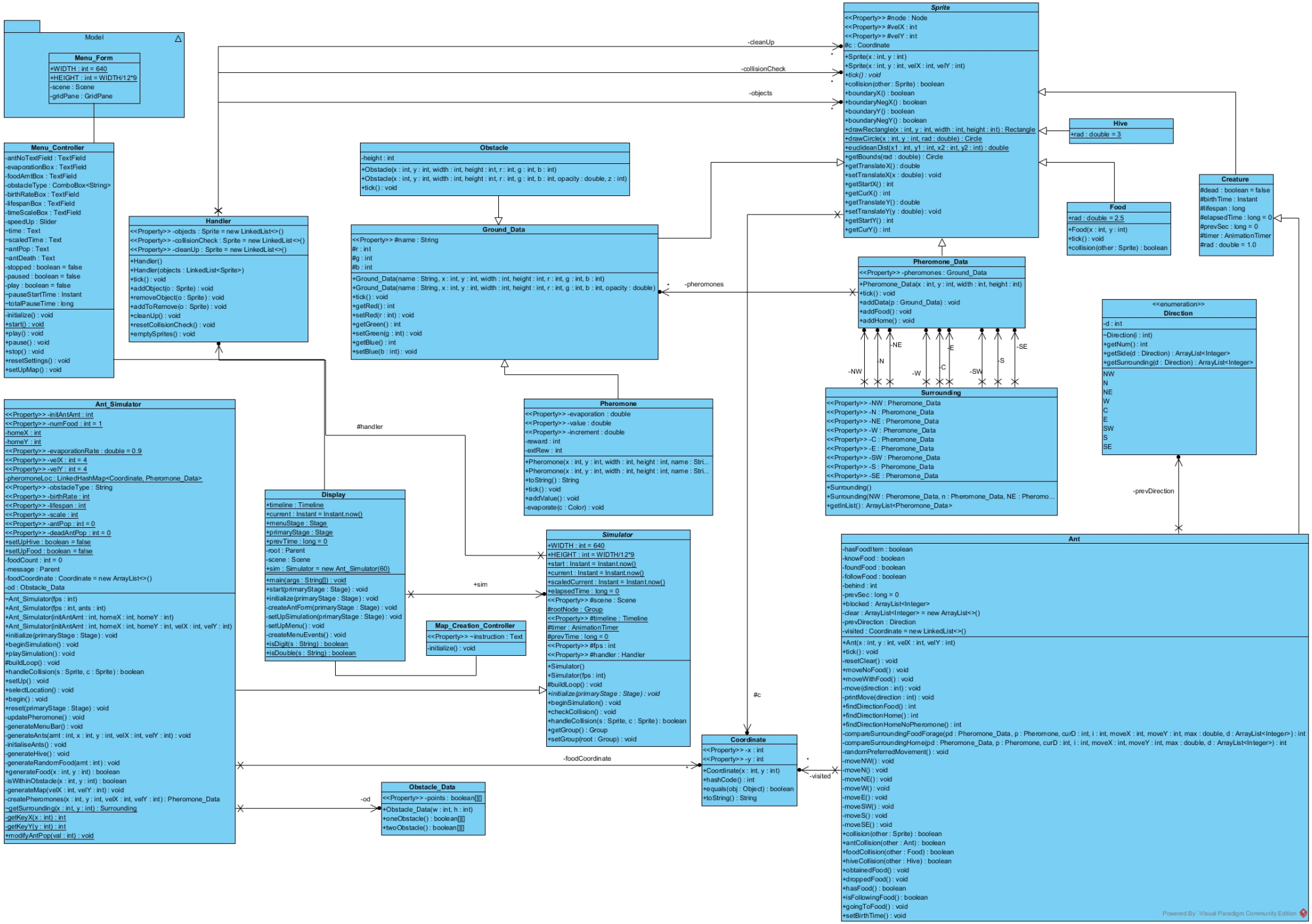

Figure 6.1: Full class diagram of the Ant simulator