**qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| SOFT152  Report  Zak Robinson - 10500227  Joe Kellaway - 10503639 |

**Tables of contents**

**Introduction**

Page 2

**UML diagram**

Page 2

**Classes and Methods**

Page 2

**Data structures**

Page 4

**Extra features**

Page 4

**Evaluation**

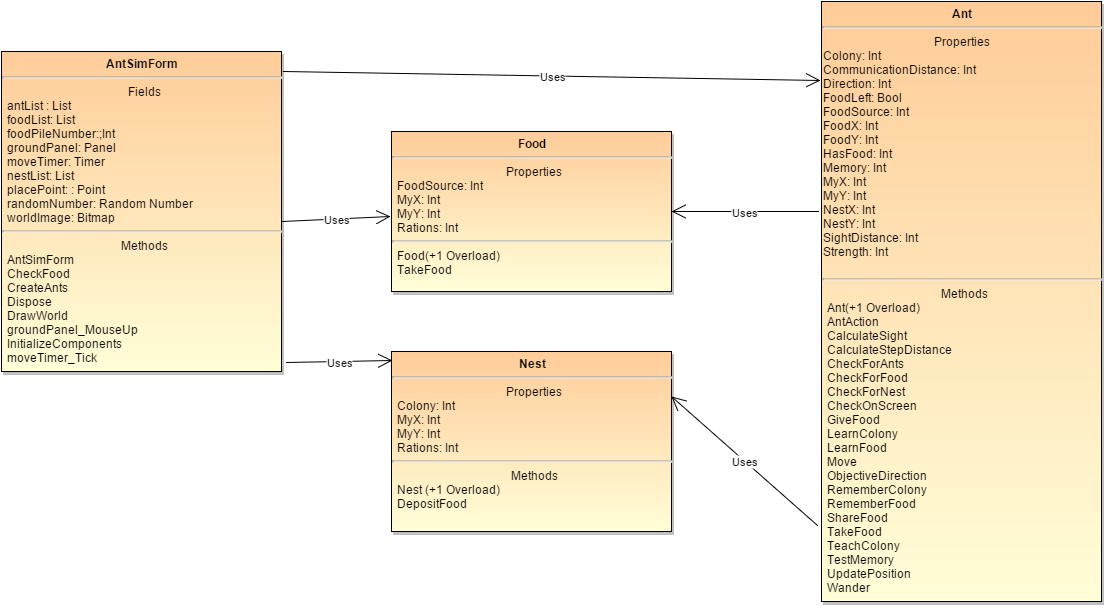
Page 4

**Possible changes**

Page 5

For this assignment we chose to complete the Ant project. We chose this project because the idea of “emerging behaviours” and being able to see this visually emerge in front of us was very exciting. There was also a lot of opportunity for extra functionalities, albeit not many that we could use to show case skills that weren’t needed to fulfil the core functionalities.

**UML Diagram for our project:**



**Ant**

The Ant class controls the behaviour of each ant, including their sight, movement, interactions and decision making.

**Methods**

* Ant(+1 overload)
* AntAction
* CalculateSightDistance
* CalculateStepDistance
* CheckForAnts
* CheckForFood
* CheckForNest
* CheckOnScreen
* GiveFood
* LearnColony
* Move
* ObjectiveDirection
* RememberColony
* RememberFood
* ShareFood
* TakeFood
* TeachColony
* TeachFood
* TestMemory
* UpdatePosition
* Wander

**Nest**

The Nest class stores information about its own location, colony number and ration count. It also allows other entities to access it for use in their own methods.

**Methods**

* Nest(+1 overload)
* DepositFood

**Food**

The Food class stores information about its own location, ration count and its foodsource number. It also allows other entities to access it for use in their own methods.

**Methods**

* Food(+1 overload)
* TakeFood

**AntSimForm**

The AntSimForm class creates the ants on-load, nests and food on-click and uses the timer to handle on-going events in the simulation.

**Methods**

* AntSimForm
* CheckFood
* CreateAnts
* DrawWorld
* groundPanel\_MouseUp
* moveTimer\_Tick
* InitializeComponents
* Dispose

**Data structures**

Our program uses two types of data structures to manage and structure the information stored about the Ants we create.

Lists – We used lists to store all of our ants, food and nests. The list holds all instances of the class Ant and allows us to generate and manage large numbers of Ants into once place, and to process or change their properties effectively using control statements created for use with lists.

Classes – We used classes to store all of the information about the individual ants in the list, including their location information, knowledge of food and nests as well as how much food they are carrying.

**Extra Features**

Extra features we included in our program that were not in the original assignment criteria

**Double Buffering**

* We implemented double buffering to fix a bug where the computer processing and drawing the list of ants took slightly too long and caused a flash as the ants were drawn and redrawn into new locations. Double buffering stores an image ready to be pasted in place of an actively changing panel.

**Evaluation**

We completed the project, doing all of what we considered were the core functions. The ant’s wander around, store information, pick up and drop of food, travel the most efficient route and most important of all communicate with each other the information that they gather on their travels. They also process and make decisions based on that information, like going to the food that is closest to their nest or learning that the food they are moving towards to is depleted.

There is a short list of improvements that we would have liked to make, most of them are quality changes rather than functional changes but we still consider them important to mention.

* + If we had time to spare we would have used imported sprites for our nests, ants and food piles. We also would have coloured them differently based on what nest they belonged to, or if they were holding food.
  + Minimising places all ants at the co-ordinate 0,0. While this isn’t a “bug” but more a product of the movement it isn’t as we intended.
  + The brief mentions robber ants as extra credit and we were unable to implement this functionality within the time provided. We gave it quite a lot of thought and decided that if we did do it we would use a Boolean value and code two movement rules into the ant class to determine what kind of ant it was. If it saw another ant with food then it would move after it, or into its path and take it before moving to its nest. For this we would have made the ants move faster when not holding food, but slower when moving with food.
  + Currently we have the ants detecting in a diamond shape around them, ideally we would have liked to have changed this to a cone shaped view. This would have been pretty simple as the ants already have a “facing” direction.
  + The collision detection for collecting and delivering food is not perfect because object data is at specific co-ordinates, not entire sprite.
  + Resizing the form caused issues so we have a set form size, it’s also easier to judge the amount of ants needed for a simulation to work correctly if the form size is locked.
  + We would have liked ants to decide whether to take a shortcut by going off the edge of the panel if it is a quicker route. We could have done this by judging its distance and if it’s over half the screen making it move in the opposite direction a set number of steps. Once the ant crossed the border then it would recalculate the distance to under half of the screen size.

We are particularly proud of the way we resolved an issue where ants would teach each other the location of depleted food.

**Possible Changes**

If we were to complete the task again there are few things we would do differently in relation to the code itself, other than do more. If we implemented the robber class of ants we would have probably liked to implement it as a sub-class of ant. This would have allowed us to use the master class of ant to create other castes of ant and create more behaviours.

We also would have liked to have a creator class for creating the ants themselves. This once again would have been important if we had implemented the ants and the robber ants in the way we stated above.