Part 1:

what you did: For this I created two classes, food and pesticide that both extended off of genericMovement. The extension was done as it already would randomly place the object and allow me to ignore placing the objects. In the constructor for the object it just sets it to stay so that it can't move.

Then within each of these objects they were given a variable holding their height. This is then calculated using the given formula and the value is stored in the variable. It was stored here because storing it in the point caused a few issues regarding the distance between two points. So for this reason the current location does not contain the y value.

how well it worked: This worked out pretty well. The food and pesticide were constantly placed at random locations within the world and given their proper calculated heights.

what its pros and cons are: The pros of this is that it's very simple. The cons is that it takes up more memory for this object than is necessary.

what you might improve: The way I can think to improve this would be to make the food and pesticide their own objects that only have the minimal amount of functionality.

Part 2:

what you did: For this I created an ant object extended from the genericMovement This object has it's own variable similar to the food and pesticide that holds its height for the purpose of simplifying calculations and movement. This height is initialized with an initHeight function that is passed the food and pesticide.

Every frame the Ant's checkDesire function will be called and passed the food and pesticide. The Ant then creates a new point and places it one step ahead of it. The food and pesticide each have their raise functions called and are passed the distance between themselves and the point. This function solves how much that point should be raised as a result of them. This is then compared to the height of the ant. If this height is greater the the angle that this point was rotated is recorded as optimal. The point is then rotated 30 degrees and the cycle repeats until the point has been rotated all around the Ant.

The turn of the ant will then be set to that of the optimal angle. With how my code is currently set up the Ant will then proceed to turn towards the optimal point, missing it but moving towards it none the less. The optimal angle will only be set for the ant if it isn't currently turning. Additionally code is in place so that the ant will instead of turning and going to unoptimal positions it will pause in place and pivot towards the optimal angle before moving.

If the Ant can't move to an optimal position it will then move to the most least optimal position.

how well it worked: This worked out quite well. The Ant almost always moved and would almost never get stuck. On a few occasions it would be unable to figure out which location it should be moving towards and would get stuck in a strange cycle of going back and forth with no knowledge of where to go.

what its pros and cons are: The pros of this system is that it's simple and effective. It also required minimal coding. the cons of this system is that in some ways it doesn't

completely fulfil the requirements where the ant doesn't move by agent movement, although its general course of movement is chosen by what gives it the best movement option.

what you might improve: I think I would improve the range that the ant checks around it. As with 30 degrees it can miss some opportunities, but at the same time it checks most of the options without requiring a huge amount of computation.