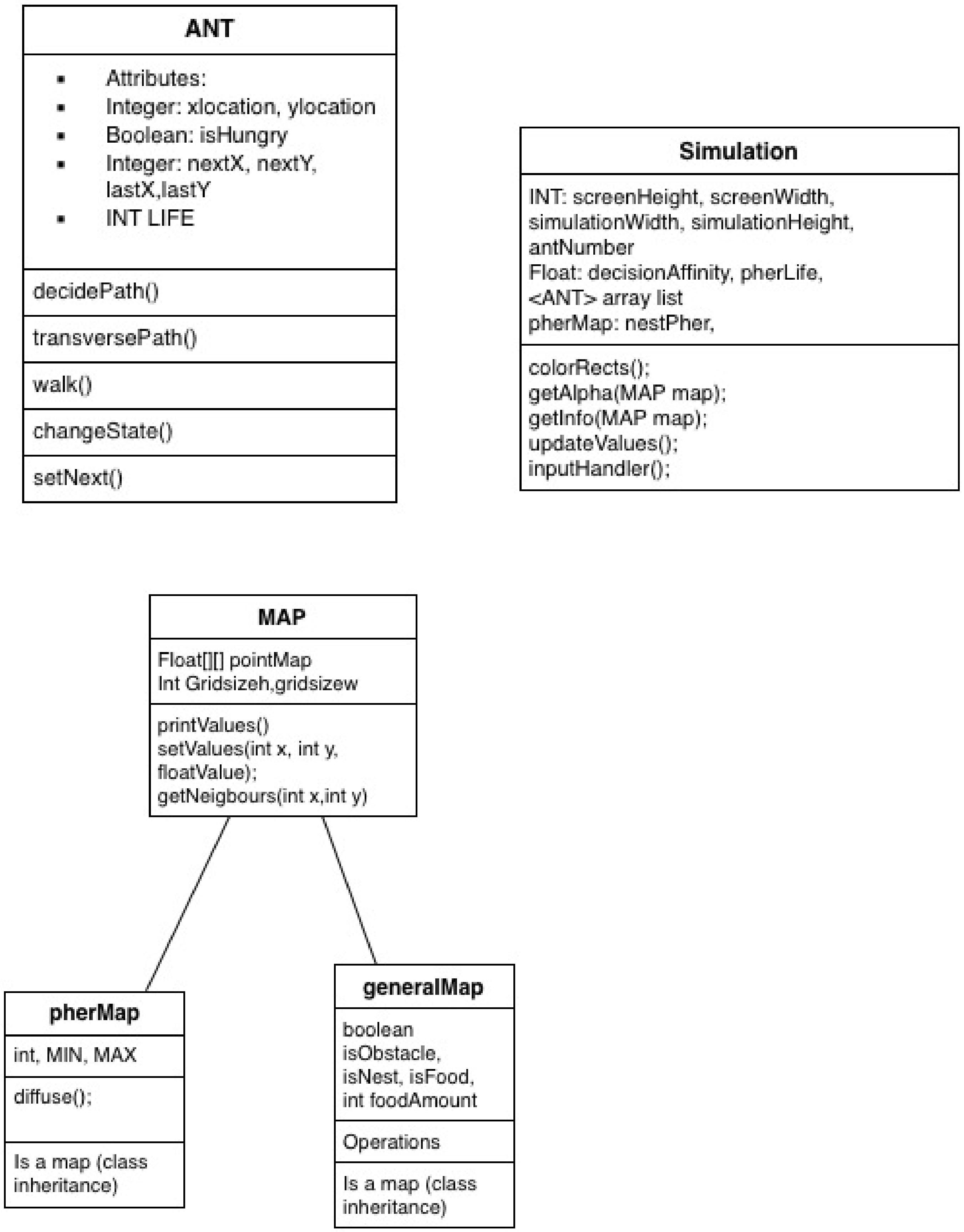
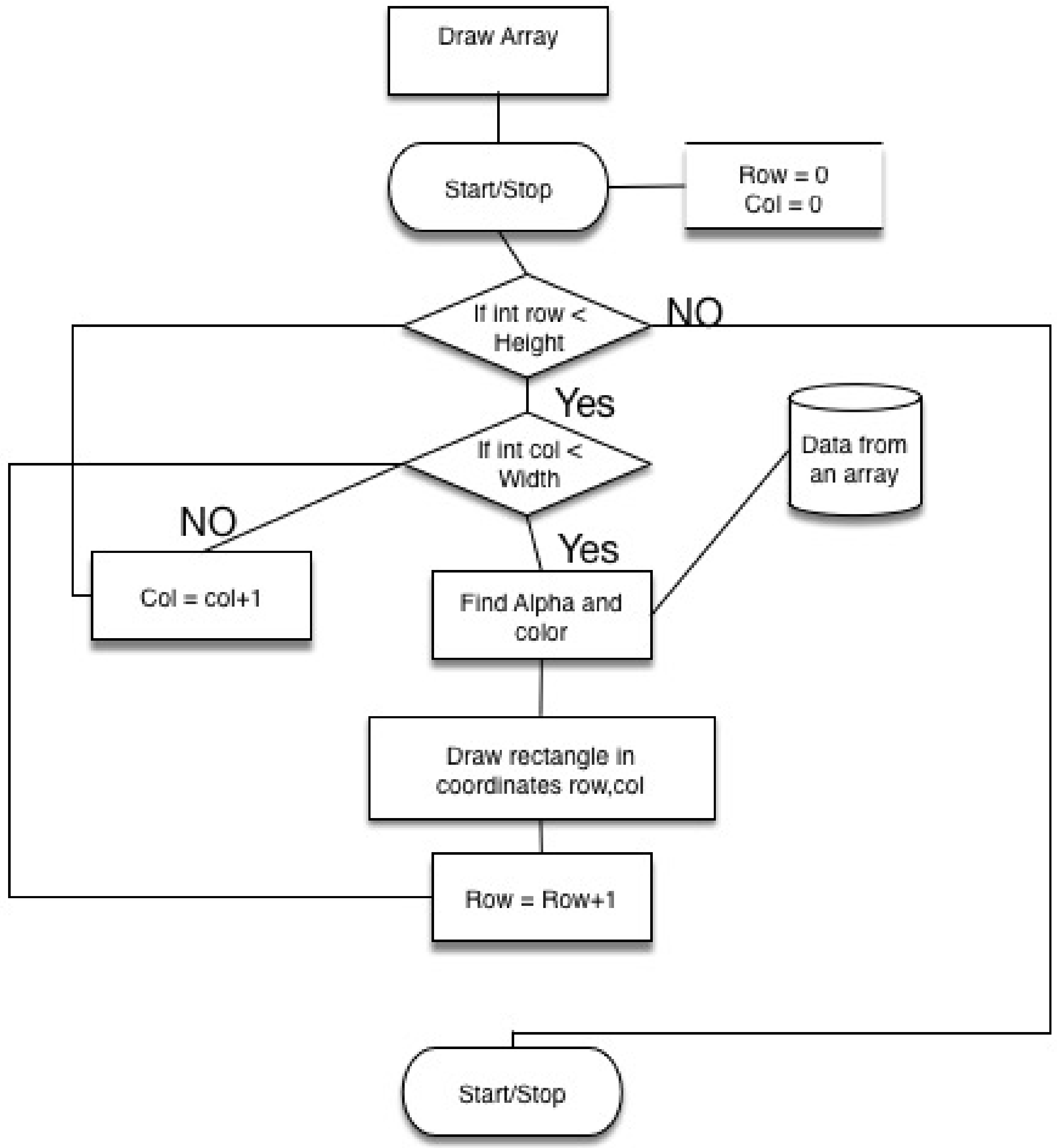
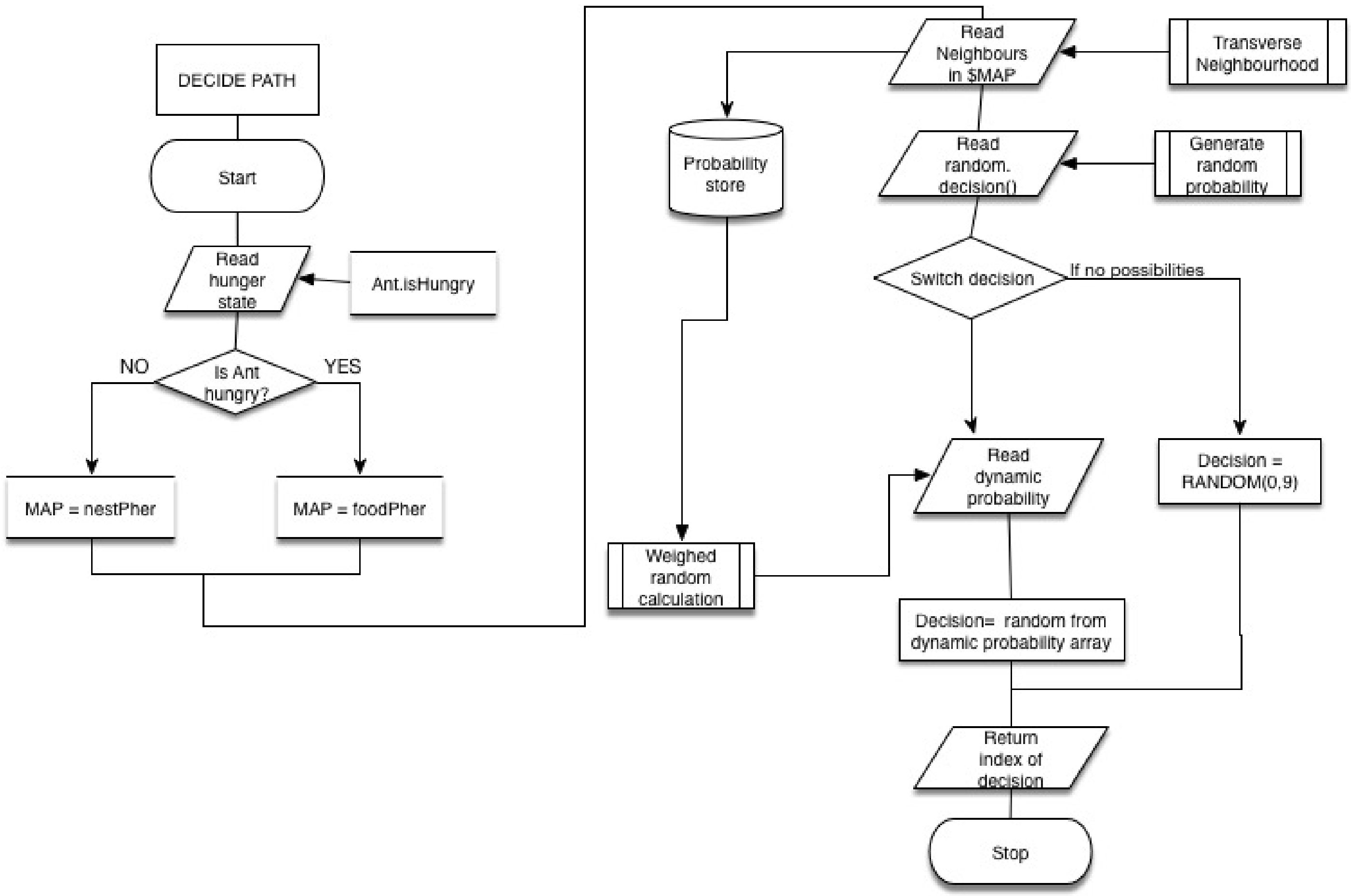
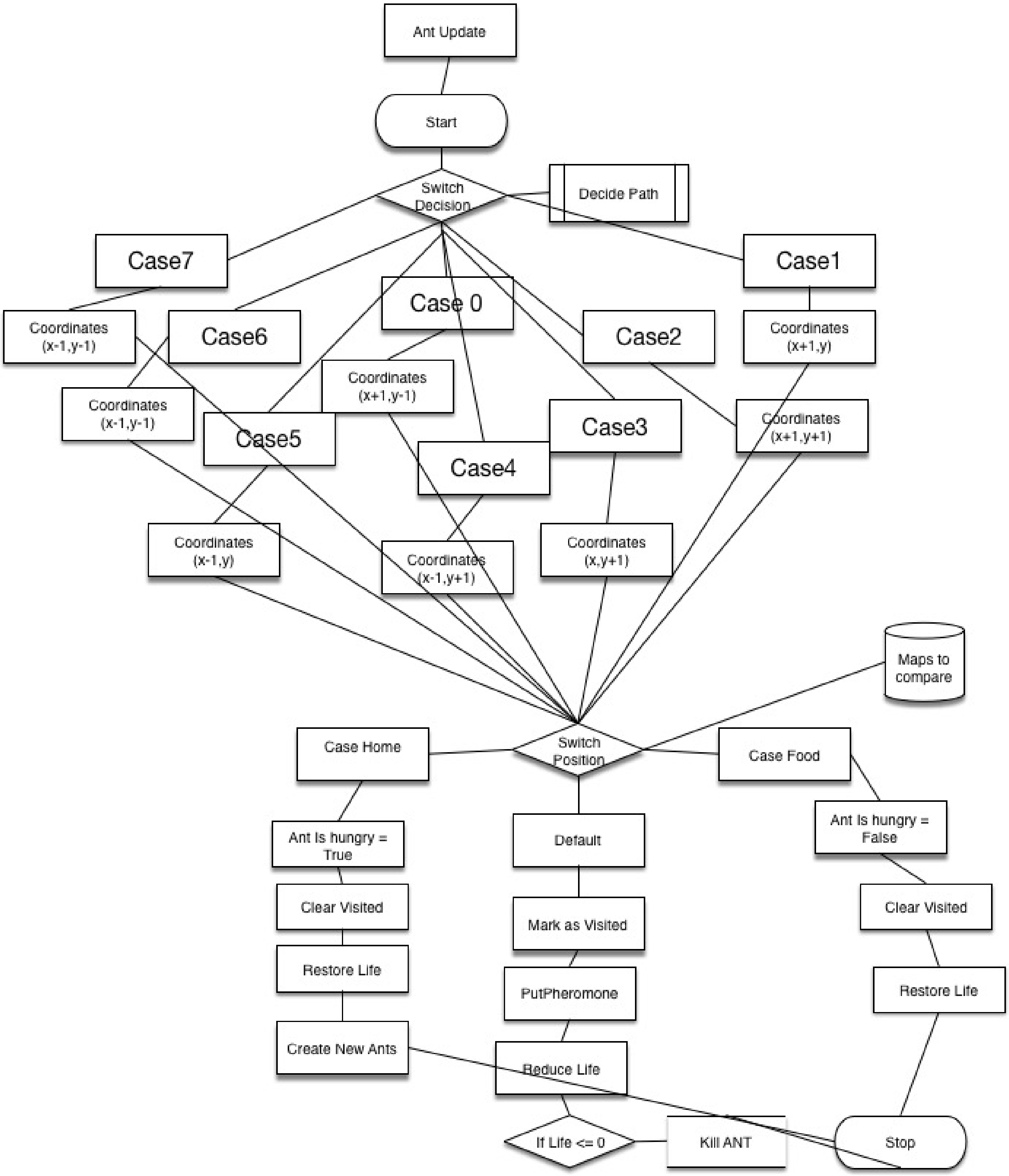

ANT
Attributes:
Integer: xlocation, ylocation
Boolean: isHungry
Integer: nextX, nextY, lastX,lastY
INT LIFE
decidePath()
transversePath()
walk()
changeState()
setNext()
Simulation
INT: screenHeight, screenWidth, simulationWidth, simulationHeight, antNumber
Float: decisionAffinity, pherLife,
<ANT> array list
pherMap: nestPher,
colorRects();
getAlpha(MAP map);
getInfo(MAP map);
updateValues();
inputHandler();
MAP
Float[][] pointMap
Int Gridsizeh,gridsizew
printValues()
setValues(int x, int y, floatValue);
getNeigbours(int x,int y)
pherMap
int, MIN, MAX
diffuse();
Is a map (class inheritance)
generalMap
boolean isObstacle, isNest, isFood, int foodAmount
Operations
Is a map (class inheritance)


Draw Array
Start/Stop
Row = 0
Col = 0
If int row < Height
NO
Yes
If int col < Width
Data from an array
NO
Yes
Col = col+1
Find Alpha and color
Draw rectangle in coordinates row,col
Row = Row+1
Start/Stop


DECIDE PATH
Start
Read hunger state
Ant.isHungry
Is Ant hungry?
NO
YES
MAP = nestPher
MAP = foodPher
Read Neighbours in $MAP
Transverse Neighbourhood
Probability store
Read random. decision()
Generate random probability
Switch decision
If no possibilities
Weighed random calculation
Read dynamic probability
Decision = RANDOM(0,9)
Decision= random from dynamic probability array
Return index of decision
Stop


Ant Update
Start
Switch Decision
Decide Path
Case7
Case1
Case 0
Coordinates (x-1,y-1)
Case6
Case2
Coordinates (x+1,y)
Coordinates (x+1,y-1)
Coordinates (x-1,y-1)
Case5
Case3
Coordinates (x+1,y+1)
Case4
Coordinates (x-1,y)
Coordinates (x-1,y+1)
Coordinates (x,y+1)
Maps to compare
Case Home
Switch Position
Case Food
Ant Is hungry = True
Default
Ant Is hungry = False
Clear Visited
Mark as Visited
Clear Visited
Restore Life
PutPheromone
Restore Life
Create New Ants
Reduce Life
If Life <= 0
Kill ANT
Stop

Actions to test	Method of testing
Ants converge on efficient path (the main algorithm).	Letting the simulation run
Ants avoid obstacles.	Placing obstacles and observing ants react by choosing other route
The user is able to interact with the simulation by dropping objects.	Ants respond to dropped stimuli, such as food, obstacles or pheromones. The objects placed by the user appear on the map
The number of ants is reasonable as well as its born/death rate.	Given enough time, and sufficient food ants do not disappear or fill the screen
The graphical representation is relevant and updated correctly.	Check that the level of pheromones and ants in a given space, corresponds to its transparency or intensity.
Check edge cases	If no food is available, the simulation should either end gracefully or continue without crashing.
Objects interact with each other only in expected ways	Making sure that pheromones, ants or obstacles in a given space do not produce unexpected results, such as crashing the simulation
The probability distribution works adequately	Isolating the code and testing it with dummy inputs with known outputs.