# Classes

Need a description of what each class does an what it inherits

Why are ants and nests so similar (A: split for semantic reasons)

What is this.sleep is it the ant sleeping?

Note: paramters of the function are appended with **[param]**

|  |  |
| --- | --- |
| Type | Description |
| integer | A whole number |
| number | A real number |
| string | A list of characters |
| Boolean | A true/1 or false/0 |
| \* object | An object |

## Ant

| **Function name** | **Variables** | **Type** | **Description** |
| --- | --- | --- | --- |
| Ant | this.size | width : integer, height : integer | The size of the ant in pixels (default: CELL\_SIZE) |
| this.coord | x : number, y : number | The coordinate of the ant |
| this.id | integer | The unique ant id |
| this.species | Species object | The ants species which determines its characteristics |
| this.type | integer | The type of ant, use the ANT\_TYPE object when setting. e.g. ANT\_TYPE.worker |
| this.nest | Nest object | The ants home nest where it was born |
| this.colour | string | The hexadecimal colour of the ant e.g. ‘#FF0000’ (default: ‘#1C1C1C’) |
| this.health | number | The health of the ant, if <= 0 the ant is dead |
| this.hungerThreshold | number | The value of health bellow which the ant is determined to be hungry (default: 100) |
| this.healthRate | number | The rate at which the ants health decreases per tick (default: 0.1) |
| this.alive | boolean | Represents the ants living state (default: true) |
| this.goal | integer | The current goal which the ant is trying to accomplish, use the GOAL object when setting e.g. GOAL.findFood (default: GOAL.none) |
| this.target | x : number, y : number | The coordinate of a target the ant has chosen, the type of target depends on the ant i.e. a worker ant targets food while soldier ants target enemy ants (default: void(0)) |
| this.itemsInView | ants: [Ant object], food : [Food object] | Holds arrays of all ants and food within view (default: ants: [], food: []) |
| this.pheromonesInRange | [Pheromone object] | An array of all the pheromones in the ants antenna range (default: []) |
| this.sleep | integer | The number of ticks the ant needs to sleep for. Used for tasks which require actions which take multiple ticks to complete (default: 0) |
| this.followingPheromone | boolean | Used to tell if an ant is following a pheromone (default: false) |
| this.direction | number | The angle which the ant is facing in radians from the vertical axis clockwise (default: \*random direction\*) |
| this.prioritizeDirection | number | The direction the ant will tend to move in, used to achieve straighter more realistic paths (default: \*random direction\*) |
| Ant.addToMap | N/A |  | Adds the current position of the ant on the map |
| Ant.removeFromMap | N/A |  | Removes the current position of the ant from the map |
| Ant.isHungry | N/A |  | Determines whether the ant is hungry or not.  **Return** boolean - true if the ant is hungry else false |
| Ant.updateSleep | N/A |  | Updates this.sleep variable to simulate time passing during sleep |
| Ant.isFood | **[param]** food | Food object | Determines whether a piece of food exists or not  **Return** boolean - true if piece of food exists else false |
| Ant.takeFood | **[param]** food | Food.object | Takes a single piece of food  **Return** boolean - true if there is still food left, else false |
| Ant.atNest | N/A |  | Determines whether the ant is currently at its own nest i.e. standing on top of a NestPiece  **Return** boolean - true if the ant is standing on its nest, else false |
| Ant.seeNest | N/A |  | Determines whether the ant can see its own nest  **Return** boolean - true if ant can see its nest, else false |
| Ant.findFoodTarget | N/A |  | Choose the target piece of food the ant should go for |
| Ant.getFood | N/A |  | Walk towards food until on top of it and then pick it up one piece at a time |
| Ant.useFood | N/A |  | Determines the best use of food i.e. either eating the food if hungry or carrying it |
| Ant.scan | N/A |  | Looks at all blocks in front of the ant within eyesight, places all items of interest into this.itemsInView |
| Ant.smell | N/A |  | Similar to this.scan - Looks at all blocks in front of the ant within antenna size, places all pheromones into this.pheromoensInRange |
| Ant.secrete | N/A |  | Secrete pheromones |
| Ant.wonder | N/A |  | Wonder around the map, following pheromones of own species otherwise picking random directions |
| Ant.move | N/A |  | Updates the ants coordinates |
| Ant.die | N/A |  | Removes the ant from the simulation |

## Worker

Inherits Ant

| **Function name** | **Variables** | **Type** | **Description** |
| --- | --- | --- | --- |
| Worker | this.coord | x : number, y : number | The coordinate of the ant |
| this.id | integer | The unique ant id |
| this.type | integer | The type of ant, use the ANT\_TYPE object when setting. e.g. ANT\_TYPE.worker (default: ANT\_TYPE.worker) |
| this.direction | number | The angle which the ant is facing in radians from the vertical axis clockwise (default: \*random direction\*) |
| this.prioritizeDirection | number | The direction the ant will tend to move in, used to achieve straighter more realistic paths (default: \*random direction\*) |
| this.carrying | integer | The amount of food the ant is carrying (default: 0) |
| this.carryingThreshold | integer | If an ant is carrying more food then this value and cannot see any food near it, ant will return to the nest to deposit the food (default: 4) |
| Worker.canCarry | N/A |  | Determines if an ant can carry food or not  **Return** boolean - true if ant can carry food else false |
| Worker.depositeFood | N/A |  | Navigate towards the nest and deposit food at the nest |
| Worker.dropFood | N/A |  | Drop a single piece of food at the nest  **Return** boolean - True if there is no more food to drop off, else false |
| Worker.useFood | N/A |  | Determines the best use of food i.e. eating it if hungry or carrying it |
|  | index - The MAP index of the ant | integer |
|  | food - The food object which is being used | Food object |
| Worker.doTask | N/A |  | Performs the actions required to complete a task |
| Worker.updateGoal | N/A |  | Determines if a goal has been completed or not and updates the next goal for the ant |
| Worker.updateHealth | N/A |  | Updates the ants this.health variable. Differs from ant.updateHealth as allows ants to eat food they are carrying if hungry |
| Worker.draw | **[param]** ctx | Canvas context object | Draws the ant onto the canvas context |
|  | scaledCoord - The coordinate of the map scaled to pixels | integer |
| Worker.update | N/A |  | Update the ant each tick |

## Queen

Inherits Ant

| **Function name** | **Variables** | **Type** | **Description** |
| --- | --- | --- | --- |
| Queen | this.coord | x : number, y : number | The coordinate of the ant |
| this.id | integer | The unique ant id |
| this.type | integer | The type of ant, use the ANT\_TYPE object when setting. e.g. ANT\_TYPE.worker (default: ANT\_TYPE.queen) |
| this.steps | integer | The number of steps the queen will take until reaching the nest site |
| Queen.doTask | N/A |  | Decide what actions need to be done to accomplish a task |
| Queen.updateGoal | N/A |  | Checks to see if the goal is accomplished and updates it if necessary |
| Queen.pickDirection | N/A |  | Used to pick a direction in which the Queen will walk a specific number of steps in (this.steps) and then create a nest |
| Queen.createNest | N/A |  | Creates a nest object |
|  | nest - The MAP index of the ant | Nest object |
|  | index - The index of the queen ant in this.species.ants used to determine which ant to remove from the list | integer |
| Queen.draw | **[param]** ctx – The canvas context | Canvas context object | Draw the queen onto the canvas context |
|  | scaledCoord - The scaled MAP coordinate to pixels | integer |
| Queen.update | N/A |  | Update the queen each tick |

## Soldier

Inherits Ant

| **Function name** | **Variables** | **Type** | **Description** |
| --- | --- | --- | --- |
| Soldier | this.coord | x : number, y : number | The coordinate of the ant |
| this.id | integer | The unique ant id |
| this.type | integer | The type of ant, use the ANT\_TYPE object when setting. e.g. ANT\_TYPE.worker (default: ANT\_TYPE.soldier) |
| this.direction | number | The direction in radians from the vertical axis clockwise (default \*random direction\*) |
| this.prioritizeDirection | number | The direction the ant will tend to move in, used to achieve straighter more realistic paths (default: \*random direction\*) |
| this.targetAnt | Ant object | The enemy ant which the soldier is targeting to attack (default: void(0)) |
| this.moving | boolean | Used for determining when an ant is in a static position e.g. guarding the nest (default: false) |
| this.steps | integer | Used for moving an ant a certain number of steps away from an object. |
| this.nearNest | boolean | Determine if an ant is close to the nest (even if it is not in view) (default: false) |
| this.nearFood | boolean | Determine if an ant is close to food (even if it is not in view) (default: false) |
| Soldier.soldiersInView | N/A |  | Determines if there are other friendly soldiers in view  **Return** boolean – true if there are soldiers in view else false |
| Soldier.seeFood | N/A |  | Determines if an ant can see food  **Return** boolean – true if the ant can see food else false |
| Soldier.pickTarget | N/A |  | Chooses which ant the soldier should target |
| Soldier.follow | N/A |  | Sets the ants direction to intercept the path of the targeted ant so that it can attack. |
| Soldier.attack | N/A |  | Attack a the targeted ant if in range |
|  | dist - The distance between the ant and its target | number |
| Soldier.updateHealth | N/A |  | Updates the soldier ants health differs from Ant.updateHealth as will change soldiers goal if hungry |
| Soldier.updateSteps | N/A |  | Updates the this.steps variable |
| Soldier.guardNest | N/A |  | Controls soldiers logic if assigned goal of guarding the nest |
| Soldier.guardPheromone | N/A |  | Controls soldiers logic if assigned goal of guarding pheromone trials |
| Soldier.guardFood | N/A |  | Controls soldiers logic if assigned goal of guarding food |
| Soldier.findFood | N/A |  | Controls soldiers logic if assigned goal of finding food e.g. if the ant is hungry |
| Soldier.doTask | N/A |  | Performs the actions required to complete a task |
| Soldier.updateGoal | N/A |  | Determines if a goal has been completed or not and updates the next goal for the ant |
| Soldier.draw | **[param]** ctx – The canvas context | Canvas context object | Draw the ant onto the canvas context |
| scaledCoord - The scaled MAP coordinate to pixels | integer |
| Soldier.update | N/A |  | Update the ant each tick |

## Species

| **Function name** | **Variables** | **Type** | **Description** |
| --- | --- | --- | --- |
| Species | this.id | integer | The unique ant id |
| this.ants | [Ant object] | Array of all ants which belong to the species (default: []) |
| this.nests | [Nest object] | Array of all nests which belong to the species (default: []) |
| this.chars.speed | number | The speed the ant moves, 1 = 1 cell per tick (default: 0.25) |
| this.chars.antennaSize | integer | The range the ant can smell pheromones (default: 5) |
| this.chars.jawStrength | integer | The strength of the ants jaw (determines how much food the ant can carry) (default: 10) |
| this.chars.jawSize | integer | The amount of damage a soldier ant does when attacking (default: 1) |
| this.chars.stingSize | integer | The range which the solider ant can attack (default: 1) |
| this.chars.eyeSight | integer | The range the ant can see items In front of it (in number of cells) (default: 5) |
| this.chars.eyeAngle | number | The angle of the sector the ant can see in front of it (default: π/2) |
| this.chars.antennaAngle | number | the angle of the sector the can can smell pheromones in front of it (default: π/2) |
| this.chars.pheromoneConcentration | number | The concentration of pheromones secreted (default: 0.4) |
| this.chars.nestCoordMemory | number | A measure of how well the ant knows where the nest is, used when navigating to the nest, represents memory of familiar landmarks near the nest (default: 0.1) |
| this.chars.explorativeInfluence | number | The likelihood of an ant changing direction rather than continuing going the direction its facing (default: 0.05) |
| this.chars.pheromoneInfluence | number | How likely it is that an ant will follow a pheromone (default: 0.9) |
| this.chars.queenStepsMin, this.chars.queenStepsMax | number | The range of steps the queen will take when navigating to a new nest site i.e. lower values mean closer nests (default: min: 200, max: 800) |
| this.chars.reproduceWorkerProb, this.chars.reproduceQueenProb, this.chars.reproduceSoldierProb | number | The probability a particular type of ant will be born compared with others (default: worker: 0.5, queen: 0.05, soldier: 0.1) Note: gets normalized so does not need to add to one. |
| this.chars.reproduceWorkerFoodCost, this.chars.reproduceQueenFoodCost, this.chars.reproduceSoldierFoodCost | number | The amount of food required to create this type of ant (This is the amount of food the ant will start with when born) (default: worker: 5, queen: 25, solider: 8) |
| this.chars.reprodcutionRate | number | The chance each tick of creating a new ant (default: 0.05) |
| this.colour.worker, this.colour.soldier, this.colour.queen, this.colour.nest, this.colour.pheromone | string | The hexadecimal colours given to each type of ant/nest/pheromone. Note: often all values are the same, (default: ‘#1C1C1C’ for all) |
| this.murationRate | number | The chance that a characteristic will be mutated |
| Species.mutateChar | **[param]** characteristic | string | Mutate a specific characteristic  **Return** number – The mutated value of the characteristic |
| Species.mutate | N/A |  | Mutate a single characteristic in the species  **Return** Species object – The new mutated species |
| altChars - The current characteristics of the species | string literal |
| altCharacterisitc - The random characteristic which will be mutated | string |
| altValue - The new value of the altCharacterisitc | variable depending on characteristic being mutated |
| species - The species new species | Species object |
| Species.createSpecies | **[param]** chars | string literal | The characteristic set of the new species  **return** Species object – The new species |
| species - The new species | Species object |
| colour - The colour of the new species | string |

## Nest

| **Function name** | **Variables** | **Type** | **Description** |
| --- | --- | --- | --- |
| **Nest** | this.nestSize | x: integer, y: integer | The number of pieces the nest extends in both the x and the y directions (default: NEST\_SIZE) |
| this.coord | x: number, y: number | The coordinate of the nest |
| this.id | integer | A unique identifier |
| this.species | Species object | The species which the nest belongs to |
| this.pieces | [NestPiece object] | An array of the NestPiece objects belonging to the nest (default: []) |
| this.health | number | The health the nest has (default: 10000) |
| this.hungerThreshold | number | The threshold below which the nest is hungry and tries to preserve food (default: 1000) |
| this.healthRate | number | The rate at which the nests health reduces each tick (default: 0.1) |
| this.alive | boolean | If nest is alive or not, needed if nest dies mid execution so does not keep acting as if it is alive for the rest of the tick (default: true) |
| **Nest.addNestPiece** | [param] coord | x: number, y: number | Create a single nest piece |
| **Nest.createNest** | N/A |  | Creates all the nest pieces |
| **Nest.die** | N/A |  | Kills the nest so that it is no longer updated and is removed from the map |
| **Nest.getCost** | [param] type – The ant type | integer – use the ANT\_TYPE object when defining | Returns the cost in the amount of health needed to create a specific type of ant  Return number – the health required |
| **Nest.calcSpeciesCost** | N/A |  | Calculates the cost of the characteristics due the spices  Return number – the health cost |
| **Nest.viable** | [param] type – The ant type | integer – use the ANT\_TYPE object when defining | Determine whether or not it is viable to create a specific type of ant  Return boolean – true if it is viable else false |
| **Nest.createAnt** | [param] type – the ant type | integer – use the ANT\_TYPE object when defining | Create a new ant |
| cost - The health cost of creating the type of ant | number |
| **Nest.reproduce** | N/A |  | Determines what type of ant to create |
| prob - A random number which will be used to determine what type of ant is created | number |
| chars - The characteristics of the species | number |
| sum - The sum of ant probabilities | number |
| queenProb, soldierProb, workerProb - The respective normalized probabilities of each type of ant | number |
| ordered - The ordered normalized probabilities of each type of ant being created | [{prob : number, type: integer}] |
| **Nest.updateHealth** | N/A |  | Update the nests health |
| **Nest.update** | N/A |  | Updates the nest each tick |

## NestPiece

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **NestPiece** | this.size | width: number, height: number | The size the piece will be in pixels (default: CELL\_SIZE) |
| this.coord | x: number, y: number | The coordinate of the piece |
| this.nest | Nest object | The nest object which controls the piece |
| this.id | integer | A unique identifier |
| this.type | integer – use the ANT\_TYPE object when defining | The type of object the piece is, as the nest piece is essentially treated as a static ant and is stored under ant in MAP, the type is needed so ants can identify the piece (default: ANT\_TYPE.nest) |
| this.heatlh | number | The health the piece has |
| **NestPiece.addToMap** | N/A |  | Adds the piece to the map |
| **Nest.removeFromMap** | N/A |  | Removes the piece from the map |
| **Nest.die** | N/A |  | Kills the nest piece stopping it from being updated and removes it from the map |
| **Nest.draw** | [param] ctx | Canvas context object | Draws the nest piece onto the canvas context |

## Pheromone

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **Pheromone** | this.concentration | number | The concentration of the pheromone |
| this.coord | x: number, y: number | The coordinate of the pheromone |
| this.size | x: integer, y: integer | the size of the pheromone in pixels (default: CELL\_SIZE) |
| this.species | Species object | The species which the pheromone belongs to |
| **Pheromone.addToMap** | N/A |  | Adds the pheromone to the map |
| **Pheromone.removeFromMap** | N/A |  | Removes the pheromone from the map |
| **Pheromone.draw** | [param] ctx – The canvas context | Canvas context object | Draw the pheromone onto the canvas context |
| **Pheromone.update** | N/A |  | Updates the pheromones concentration each tick |

## FoodSystem

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **FoodSystem** | this.variation | min: integer, max: integer | The minimum and maximum amounts of food a single piece can contain |
|  | this.colour | string | The hexadecimal colour of the piece of food (default: FOOD\_COLOUR). Note: opacity depends on food amount |
| **FoodSystem.addFoodBlob** | [param] coord – The coordinate of the food | x: number, y: number | Creates a circle of food with reducing amounts around a coordinate. |
|  | [param] radius – The radius of the food | integer |
|  | affectedCells – The cells which lie in the food blob | [{x: integer, y: integer}] |
|  | distanceFromCenter – The distance between the cell and the coord | number |
|  | amount – The food amount | integer |
| **FoodSystem.addFood** | N/A |  | Adds random sized food blobs at random positions in the map |

## Food

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **Food** | this.foodSystem | FoodSystem object | The food system which controls the food pieces |
|  | this.size | width: number, height: number | The size of the food in pixels (default: CELL\_SIZE) |
|  | this.amount | integer | The concentration of food the piece has |
|  | this.coord | x: number, y: number | The coordinate of the piece of food |
| **Food.addToMap** | N/A |  | Adds the piece of food to the map |
| **Food.removeFromMap** | N/A |  | Removes the piece of food from the map |
| **Food.draw** | [param] ctx – The canvas context | Canvas context object | Draws the piece of food onto the canvas context |

# Modules

## Map

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **drawGrid** | [param] ctx | Canvas context object | Draws a grid onto the canvas context |
| **drawBackground** | [param] ctx | Canvas context object | Draws a rectangle over the entire canvas effectively wiping it to a single colour |
| **zoom** | [param] level – The zoom level (positive : zoom in, negative : zoom out) | number | Change the simulation zoom level |
| **createMap** | N/A |  | Create an empty map and populate it with default values |

## Canvas

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **resizeElement** | [param] element – The element to be resized | HTML element | Resizes a HTML element to a size |
| [param] size – The new size of the element | width: number, height: number |
| **drawRect** | [param] ctx – The canvas context | Canvas context object | Draws a rectangle onto a canvas context |
| [param] coord – The coordinate of the top left corner | x: number, y: number |
| [param] size – The size of the rectable | width: number, height: number |
| [param] fillColour – The colour of the rectangle | string |
| [param] strokeColour – The colour of the rectangles boarder (default: ‘#000000’) | string |
| [param] lineWidth – The width of the border (default: 0) | number |
| **drawLine** | [param] ctx – The context which the line will be drawn onto | Canvas context object | Draws a line onto a canvas context |
| [param] coord1 – The coordinate of the starting point | x: number, y: number |
| [param] coord2 – The coordinate of the end point | x: number, y: number |
| [param] strokeColour – The stroke colour of the line | string |
| [param] lineWidth – The width of the stroke | number |
| **drawArc** | [param] ctx – The canvas context which the arc will be drawn onto | Canvas context object | Draws an arc onto a canvas context |
| [param] coord – The coordinate of the centre of the arc | x: number, y: number |
| [param] radius – The radius of the arc | number |
| [param] startAngle – The angle from the horizontal to start the arc at (clockwise) | number |
| [param] endAngle – The angle from the horizontal to stop the arc at (clockwise) | number |
| [param] strokeColour – The stroke colour of the arc | string |
| [param] lineWidth – The width of the stroke | number |
| [param] fillColour – The colour of the arc | string |
| **drawCircle** | [param] ctx – The canvas context which the circle will be drawn onto | Canvas context object | Draws a circle onto a canvas context |
| [param] coord – The coordinate of the centre of the circle | x: number, y: number |  |
| [param] radius – The radius of the circle | number |  |
| [param] fillColour – The colour of the circle | string |  |
| **clearCanvas** | [param] ctx – The canvas context | Canvas context object | Draws a rectangle over the entire canvas effectively wiping it to a single colour (OUT\_OF\_BOUNDS\_COLOUR) |

## Utilities

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **randInt** | range – The range inclusive | min: integer, max: integer | Returns a random integer within a specific range  Return integer – The random number |
| **randFloat** | range – The range inclusive | min: number, max: number | Returns a random float within a specific range  Return number – The random float |
| **randDir** | N/A |  | Returns a random angle between 0 and 2π radians  Return number – The random direction |
| **randColour** | N/A |  | Returns a random hexadecimal colour  Return string – The random colour |
| **validateDirection** | dir – The direction | number | Returns an angle in the range 0 to 2π e.g. if dir = 4π returns 2π  Return number – The new angle |
| **randProperty** | obj – The object literal | object | Picks a random property of an object literal  Return object – The random property from the object |
| **distance** | coord1, coord2 – The coordinates to find the distance between | x: number, y: number | Returns the shortest distance between two coordinates  Return number – The distance |
| x – The distance in the x axis | number |
| y – The distance in the y axis | number |
| **scaleCoord** | coord – The coordinate to scale | x: number, y: number | Scale a coordinate to its location in pixels  Return {x: number, y: number} – The scaled coordinate |
| **coordToIndex** | coord – The coordinate which will be converted | x: number, y: number | Converts a coordinate to a map index, it can be used with coordinates which don’t map exactly to a single index and will work out which cell the coordinate lies in mostly  Return integer – The map index which the coordinate converts to |
| **indexToCoord** | index – The map index | integer | Converts a map index to a coordinate  Return {x: number, y: number} – The coordinate which the index converts to |
| x – The x coordinate of the index | number |
| y – The y coordinate of the index | number |
| **getCellCoord** | coord – The coordinate | x: number, y: number | Similar to coordToIndex however returns the \*neat\* coordinate i.e. a coordinate which maps exactly to a single map index by finding the cell which the coordinate lies in mostly  Return {x: number, y: number} – The coordinate which maps exactly to a single map index |
| **boundary** | coord – The coordinate to test | x: number, y: number | Returns the \*wrapped\* coordinate of a coordinate if it exceeds the map boundary. As the map wraps around if an ant foes off one side it will appear on the other  Return {x: number, y: number} – The wrapped coordinate |
| bounds – The boundary which is tested against | x: {min: number, max: number}, y: {min: number, max: number} |
| **getBlock** | coord – The coordinate to get the block around | x: number, y: number | Returns an array of cells which lie a certain distance around a specific point  Return {x: number, y: number} – An array of coordinate which lie around the coordinate |
| size – The size of the block i.e. if width = 2, takes 2 blocks to the left and two blocks on the right of the coordinate | width: integer, height: integer |
| **getSector** | coord – The coordinate to get the block around | x: number, y: number | Returns an array of cells which lie in the sector of a circle of a particular radius around a specific point  Return [{x: number, y: number}] – An array of coordinates which lie in the sector |
| radius – The radius of the sector | number |
| direction – The direction the ant is facing | number |
| angle – The angle of the sector | number |
| searchCoord – The coordinate of the cell currently being checked to see if it lies in the sector | x: number, y: number |
| **turnAround** | angle – The angle in radians | number | Returns the reverse direction  Return number – The reversed direction |
| **angleTo** | coord, target – The coordinates to find the angle between | x: number, y: number | Returns the direction/angle of shortest path to get from the coord to the target  Return number – the angle from the vertical axis clockwise in radians |
| **createAnt** | species – The species of the new ant | Species object | Creates a new ant |
| coord – The coordinate of the new ant | x: number, y: number |
| nest – The new ants home nest | Nest object |
| startingHealth – The new ants health | number |
| type – The type of ant to create e.g. ANT\_TYPE.worker | integer |
| **genID** | N/A |  | Generates a unique id. Requires CURRENT\_ID variable to keep track of current id  Return integer – A unique ID |
| **clone** | obj – The object which will be cloned | object | Clones an object (needed as JavaScript passes everything by reference)  Return object – A copy of object |
| temp – A copy of the object | object |
| **getElement** | id – The HTML id | string | Returns the HTML element on ID responds to  Return HTML element object – The HTML element |
| **setValue** | id – The HTML id | string | Sets the value of a HTML element |
| value – The value which the HTML element is being set to | string |

Controls

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Variables** | **Type** | **Description** |
| **start** | N/A |  | Starts the simulation |
| **pause** | N/A |  | Stops the simulation |
| **toggleRunning** | N/A |  | Toggles between running and paused |
| **step** | N/A |  | Steps through a single tick of the simulation |
| **runPauseButton** | N/A |  | Toggles running and updates the pause/run button |
| **updateValue** | [param] element – An element of an input in the settings panel | HTML element object | Updates the value of a characteristic i.e. when a user slides a slider, the characteristics value is updated |
| [param] value – The new value of the characteristic | string |
| characteristic – The characteristic being updated | string literal |
| speciesCost – The cost of the species with the updated value | number |
| workerFoodCost, queenFoodCost, soldierFoodCost – The costs of each type of ant | number |
| queenStepsMax, queenStepsMin – The number of queen steps (used to determine if the min is > then the max) | number |
| **updateDefaultValues** | N/A |  | Updates the value of all characteristics to their default values |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |