Ant Colony

Generated by Doxygen 1.8.6

Wed Apr 9 2014 18:43:11

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 ants Namespace Reference

Classes

- class Ant
- class WorkerAnt
- class QueenAnt

5.2 constants Namespace Reference

Variables

- tuple DIRECTIONS
- int WIDTH = 80
- int HEIGHT = 60

5.2.1 Variable Documentation

5.2.1.1 tuple constants.DIRECTIONS

Initial value:

```
00001 = ((1,0), (1,1), (0,1), (-1,1),
00002     (-1,0), (-1,-1), (0,-1), (1,-1))
```

Definition at line 1 of file constants.py.

5.2.1.2 int constants.HEIGHT = 60

Definition at line 4 of file constants.py.

5.2.1.3 int constants.WIDTH = 80

Definition at line 3 of file constants.py.

5.3 controller Namespace Reference

Classes

- class Simulation

5.4 display Namespace Reference

Classes

- class Entity

5.5 main Namespace Reference

Variables

- tuple simulation = Simulation()

5.5.1 Variable Documentation

5.5.1.1 tuple main.simulation = Simulation()

Definition at line 4 of file main.py.

5.6 task_manager Namespace Reference

Classes

- class TaskManager
- class Task
- class Explore
- class TakeFood
- class DropFood
- class FollowFoodTrail
- class FollowHomeTrail

5.7 world Namespace Reference

Classes

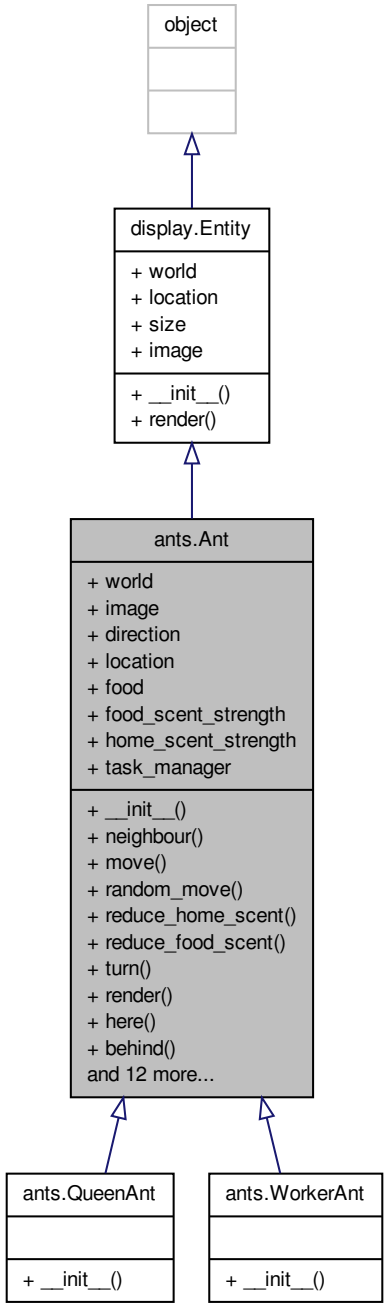
- class Cell
- class World

Chapter 6

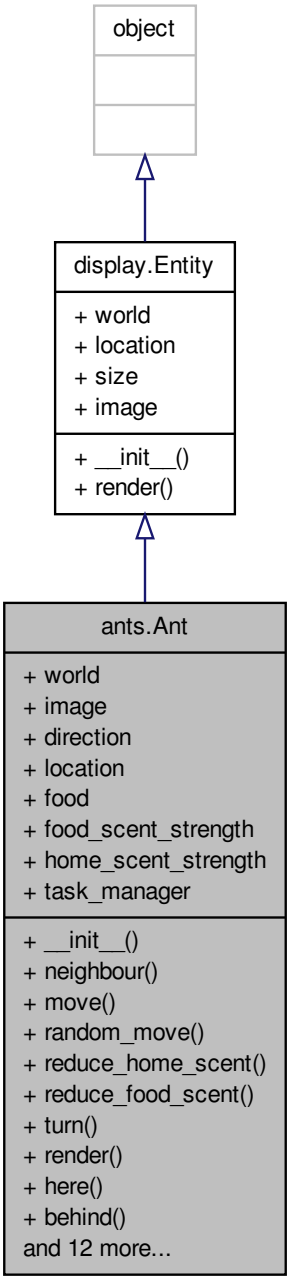
Class Documentation

6.1 ants.Ant Class Reference

Inheritance diagram for ants.Ant:



Collaboration diagram for ants.Ant:



Public Member Functions

- def __init__
- def neighbour
- def move
- def random_move
- def reduce_home_scent

- def reduce_food_scent
- def turn
- def render
- def here
- def behind
- def ahead
- def ahead_left
- def ahead_right
- def locate_food_nearby
- def locate_home_nearby
- def locate_home_scent_nearby
- def locate_food_scent_nearby
- def rank_by_food_scent
- def rank_by_home_scent
- def drop_food
- def has_food
- def __nonzero__

Public Attributes

- world
- image
- direction
- location
- food
- food_scent_strength
- home_scent_strength
- task_manager

6.1.1 Detailed Description

```
A virtual base class for Ants
```

Definition at line 6 of file ants.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 def ants.Ant.__init__ (*self, world, image, direction, location*)

Definition at line 8 of file ants.py.

6.1.3 Member Function Documentation

6.1.3.1 def ants.Ant.__nonzero__ (*self*)

Definition at line 197 of file ants.py.

6.1.3.2 def ants.Ant.ahead (*self*)

```
The cell just ahead
```

Definition at line 77 of file ants.py.

6.1.3.3 def ants.Ant.ahead_left (*self*)

```
The cell just ahead-left
```

Definition at line 81 of file ants.py.

6.1.3.4 def ants.Ant.ahead_right (*self*)

```
The cell just ahead-right
```

Definition at line 85 of file ants.py.

6.1.3.5 def ants.Ant.behind (*self*)

```
The cell just behind
```

Definition at line 73 of file ants.py.

6.1.3.6 def ants.Ant.drop_food (*self*)

```
Set food to zero
Update the food values of the home cell it reached
```

Definition at line 187 of file ants.py.

6.1.3.7 def ants.Ant.has_food (*self*)

Definition at line 194 of file ants.py.

6.1.3.8 def ants.Ant.here (*self*)

```
The cell it is standing on
```

Definition at line 69 of file ants.py.

6.1.3.9 def ants.Ant.locate_food_nearby (*self*)

```
Locate all sources nearby and return any one randomly
return None if no food source is found
```

Definition at line 89 of file ants.py.

6.1.3.10 def ants.Ant.locate_food_scent_nearby (*self*)

```
Scan the 5 directions near the direction of the ant for food scent and
return one random direction
return None if not found
```

Definition at line 140 of file ants.py.

6.1.3.11 def ants.Ant.locate_home_nearby (*self*)

```
Locate home cell nearby and return any one randomly
return None if not found
```

Definition at line 105 of file ants.py.

6.1.3.12 def ants.Ant.locate_home_scent_nearby (*self*)

```
Scan the 5 directions near the direction of the ant for home scent and
return one random direction
return None if not found
```

Definition at line 121 of file ants.py.

6.1.3.13 def ants.Ant.move (*self*)

```
Move the ant by a unit,
Leave a scent trail,
remove the ant from its old cell, and
update the current cell ant with itself
```

Definition at line 27 of file ants.py.

6.1.3.14 def ants.Ant.neighbour (*self, direction*)

```
Returns location of neighbouring cell in a direction
relative to the ant direction
```

Definition at line 20 of file ants.py.

6.1.3.15 def ants.Ant.random_move (*self*)

```
Ant makes a move forward or turns randomly
```

Definition at line 41 of file ants.py.

6.1.3.16 def ants.Ant.rank_by_food_scent (*self*)

```
Scan the 5 directions near the direction of the ant for food scent and
return the direction with the strongest scent
return None if not found
```

Definition at line 159 of file ants.py.

6.1.3.17 def ants.Ant.rank_by_home_scent (*self*)

```
Scan the 5 directions near the direction of the ant for home scent and
return the direction with the strongest scent
return None if not found
```

Definition at line 173 of file ants.py.

6.1.3.18 def ants.Ant.reduce_food_scent (*self*, *amt* = 1)

```
Reduce food scent by 'amt'
```

Definition at line 53 of file ants.py.

6.1.3.19 def ants.Ant.reduce_home_scent (*self*, *amt* = 1)

```
Reduce home scent by 'amt'
```

Definition at line 48 of file ants.py.

6.1.3.20 def ants.Ant.render (*self*)

```
Render itself
```

Definition at line 62 of file ants.py.

6.1.3.21 def ants.Ant.turn (*self*, *n*)

```
Changes direction n times
```

Definition at line 58 of file ants.py.

6.1.4 Member Data Documentation

6.1.4.1 ants.Ant.direction

Definition at line 12 of file ants.py.

6.1.4.2 ants.Ant.food

Definition at line 14 of file ants.py.

6.1.4.3 ants.Ant.food_scent_strength

Definition at line 15 of file ants.py.

6.1.4.4 ants.Ant.home_scent_strength

Definition at line 16 of file ants.py.

6.1.4.5 ants.Ant.image

Definition at line 11 of file ants.py.

6.1.4.6 ants.Ant.location

Definition at line 13 of file ants.py.

6.1.4.7 ants.Ant.task_manager

Definition at line 18 of file ants.py.

6.1.4.8 ants.Ant.world

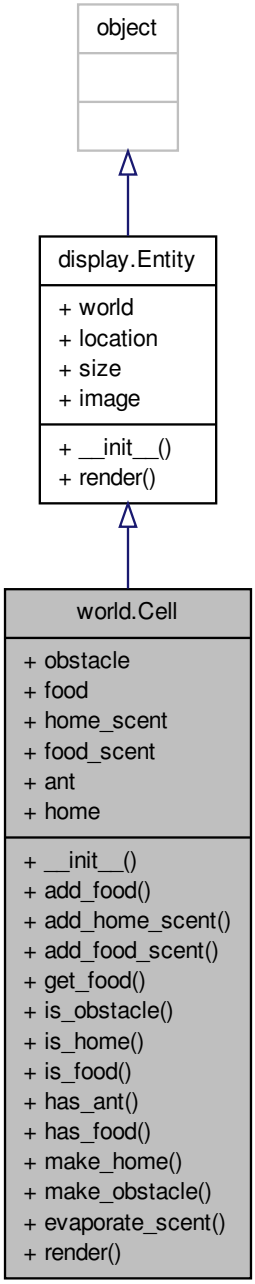
Definition at line 10 of file ants.py.

The documentation for this class was generated from the following file:

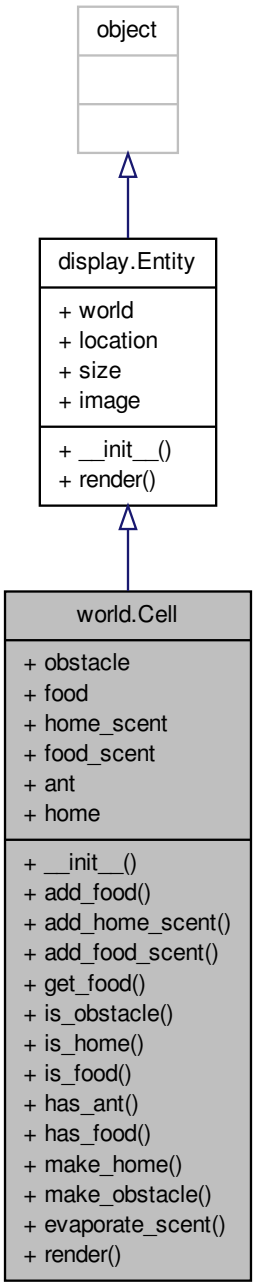
- ants.py

6.2 world.Cell Class Reference

Inheritance diagram for world.Cell:



Collaboration diagram for world.Cell:



Public Member Functions

- def __init__
- def add_food
- def add_home_scent
- def add_food_scent
- def get_food

- def is_obstacle
- def is_home
- def is_food
- def has_ant
- def has_food
- def make_home
- def make_obstacle
- def evaporate_scent
- def render

Public Attributes

- obstacle
- food
- home_scent
- food_scent
- ant
- home

6.2.1 Detailed Description

```
Data containers for each location in World
```

Definition at line 8 of file world.py.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 def world.Cell.__init__ (*self, world, i, j, cell_size*)

Definition at line 10 of file world.py.

6.2.3 Member Function Documentation

6.2.3.1 def world.Cell.add_food (*self, amt*)

Definition at line 19 of file world.py.

6.2.3.2 def world.Cell.add_food_scent (*self, amt*)

Definition at line 25 of file world.py.

6.2.3.3 def world.Cell.add_home_scent (*self, amt*)

Definition at line 22 of file world.py.

6.2.3.4 def world.Cell.evaporate_scent (*self, rate*)

```
Evaporates scent ( decay law )
```

Definition at line 59 of file world.py.

6.2.3.5 def world.Cell.get_food (*self, amt*)

```
Get "amt" amount of food if available else returns whatever food is available
```

Definition at line 28 of file world.py.

6.2.3.6 def world.Cell.has_ant (*self*)

Definition at line 47 of file world.py.

6.2.3.7 def world.Cell.has_food (*self*)

Definition at line 50 of file world.py.

6.2.3.8 def world.Cell.is_food (*self*)

Definition at line 44 of file world.py.

6.2.3.9 def world.Cell.is_home (*self*)

Definition at line 41 of file world.py.

6.2.3.10 def world.Cell.is_obstacle (*self*)

Definition at line 38 of file world.py.

6.2.3.11 def world.Cell.make_home (*self*)

Definition at line 53 of file world.py.

6.2.3.12 def world.Cell.make_obstacle (*self*)

Definition at line 56 of file world.py.

6.2.3.13 def world.Cell.render (*self*)

```
Changes "index" to render the cell according to what it represents
(home, food, etc) and calls the super class
Also renders scent levels with transparency depending on its strength
```

Definition at line 68 of file world.py.

6.2.4 Member Data Documentation

6.2.4.1 world.Cell.ant

Definition at line 15 of file world.py.

6.2.4.2 world.Cell.food

Definition at line 12 of file world.py.

6.2.4.3 world.Cell.food_scent

Definition at line 14 of file world.py.

6.2.4.4 world.Cell.home

Definition at line 16 of file world.py.

6.2.4.5 world.Cell.home_scent

Definition at line 13 of file world.py.

6.2.4.6 world.Cell.obstacle

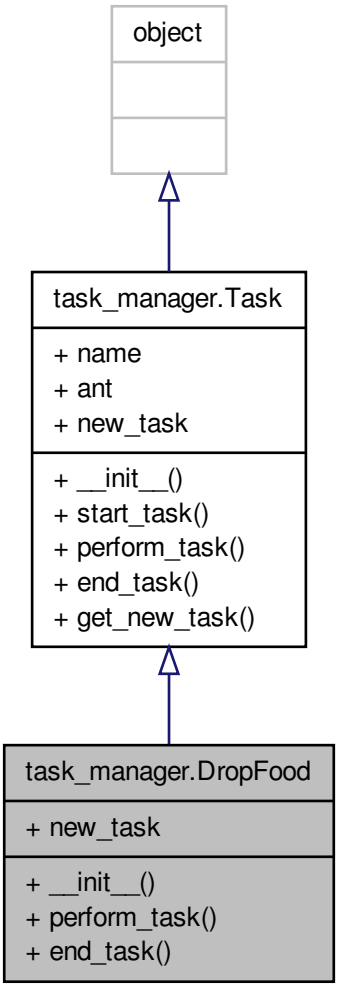
Definition at line 11 of file world.py.

The documentation for this class was generated from the following file:

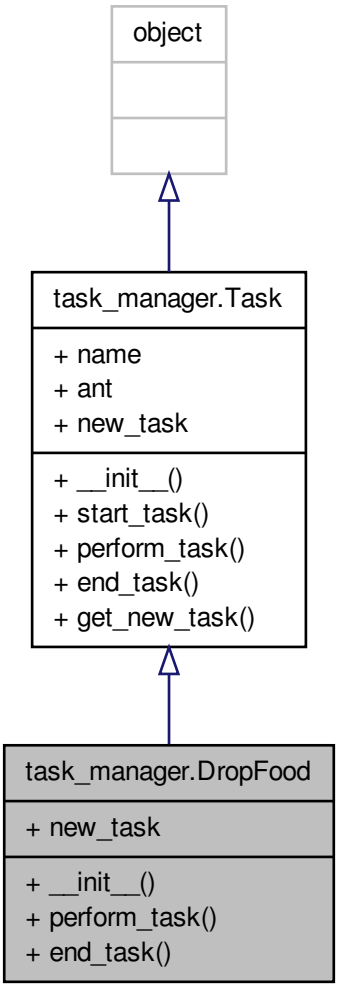
- world.py

6.3 task_manager.DropFood Class Reference

Inheritance diagram for task_manager.DropFood:



Collaboration diagram for task_manager.DropFood:



Public Member Functions

- def __init__
- def perform_task
- def end_task

Public Attributes

- new_task

6.3.1 Detailed Description

```
Drop Food Task
```

Definition at line 126 of file task_manager.py.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 def task_manager.DropFood.__init__ (*self, ant*)

Definition at line 128 of file task_manager.py.

6.3.3 Member Function Documentation

6.3.3.1 def task_manager.DropFood.end_task (*self*)

```
Increase home scent strength and reduce food scent strength
```

Definition at line 146 of file task_manager.py.

6.3.3.2 def task_manager.DropFood.perform_task (*self*)

```
If ant reaches home drop the food inside the home
otherwise follow a home trail
```

Definition at line 131 of file task_manager.py.

6.3.4 Member Data Documentation

6.3.4.1 task_manager.DropFood.new_task

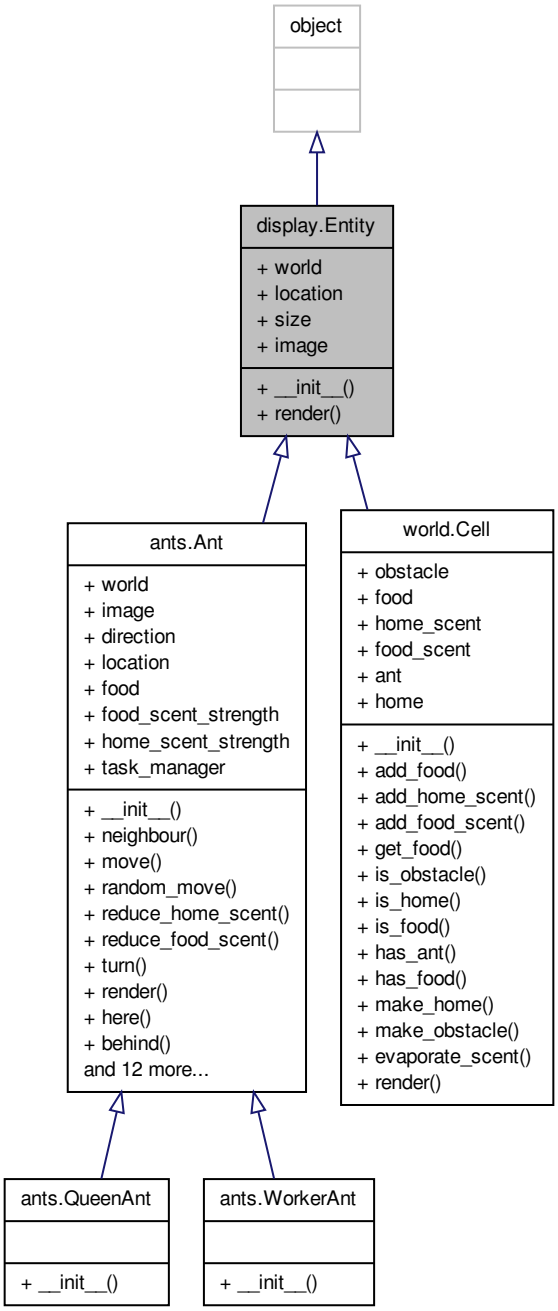
Definition at line 142 of file task_manager.py.

The documentation for this class was generated from the following file:

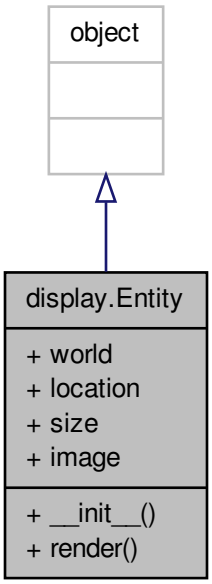
- task_manager.py

6.4 display.Entity Class Reference

Inheritance diagram for display.Entity:



Collaboration diagram for display.Entity:



Public Member Functions

- def __init__
- def render

Public Attributes

- world
- location
- size
- image

6.4.1 Detailed Description

```
Base class for all drawable objects
```

Definition at line 1 of file display.py.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 def display.Entity.__init__ (*self, world, location, size, image*)

Definition at line 3 of file display.py.

6.4.3 Member Function Documentation

6.4.3.1 def display.Entity.render (*self*, *index* = 0)

```
Draw the object into the screen
    - selects the portion of the image to draw from the "index" argument
```

Definition at line 9 of file display.py.

6.4.4 Member Data Documentation

6.4.4.1 display.Entity.image

Definition at line 7 of file display.py.

6.4.4.2 display.Entity.location

Definition at line 5 of file display.py.

6.4.4.3 display.Entity.size

Definition at line 6 of file display.py.

6.4.4.4 display.Entity.world

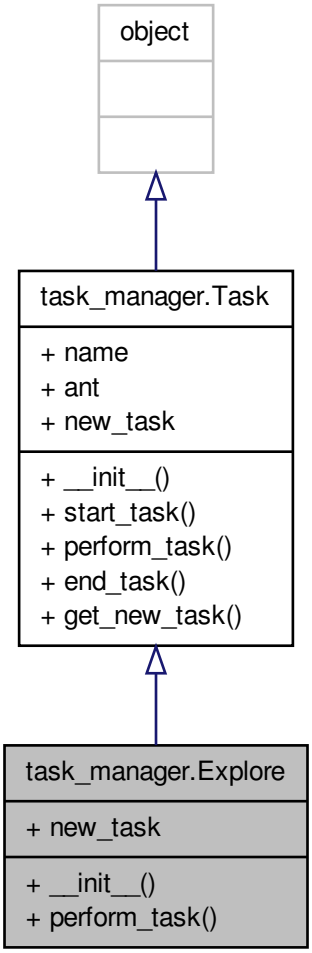
Definition at line 4 of file display.py.

The documentation for this class was generated from the following file:

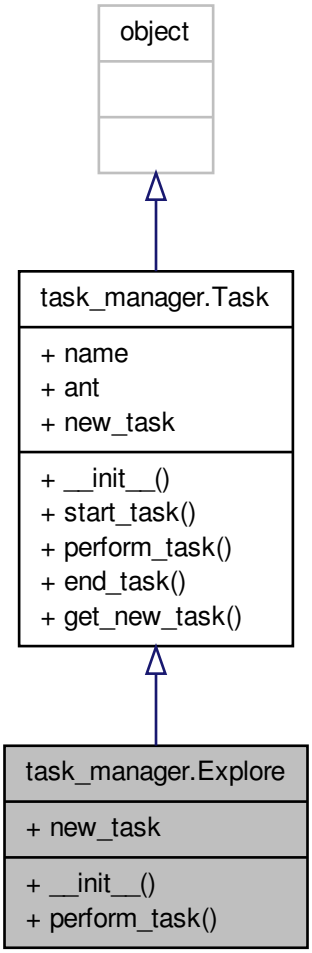
- display.py

6.5 task_manager.Explore Class Reference

Inheritance diagram for task_manager.Explore:



Collaboration diagram for task_manager.Explore:



Public Member Functions

- def __init__
- def perform_task

Public Attributes

- new_task

6.5.1 Detailed Description

```
Ant Exploring Task
```

Definition at line 56 of file task_manager.py.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 def task_manager.Explore.__init__ (*self, ant*)

Definition at line 58 of file task_manager.py.

6.5.3 Member Function Documentation

6.5.3.1 def task_manager.Explore.perform_task (*self*)

```
If ant has food -
    find home nearby and drop food there, else
    find home scent nearby and switch to that task, else
    make a random move
If ant is searching for food
    if food is found switch to take food task, else
    find a food scent trail, else
    avoid obstacles
    reverse direction if it finds home nearby
Reduce it scent strength by an unit
```

Definition at line 61 of file task_manager.py.

6.5.4 Member Data Documentation

6.5.4.1 task_manager.Explore.new_task

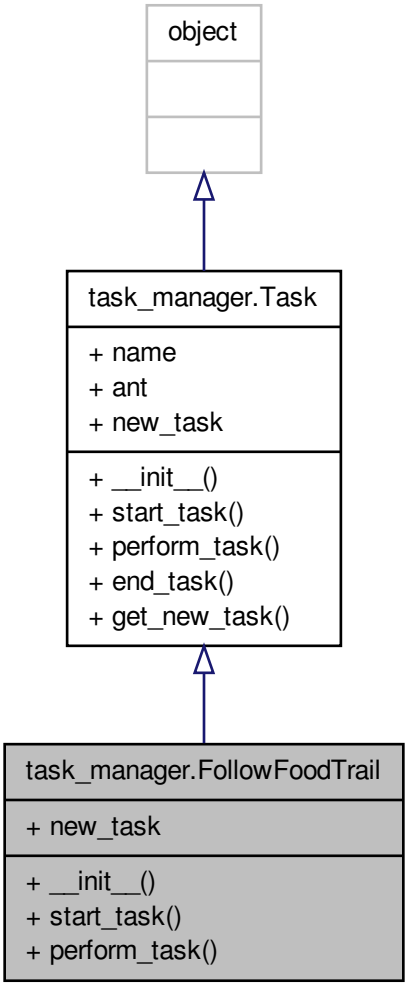
Definition at line 79 of file task_manager.py.

The documentation for this class was generated from the following file:

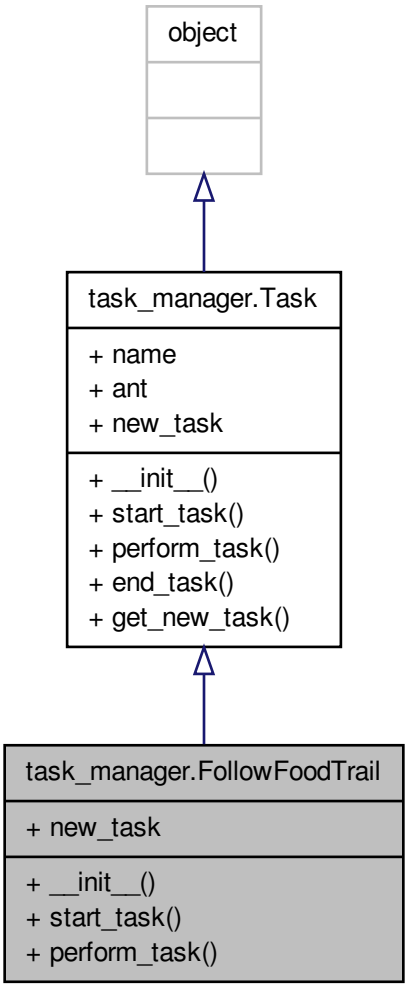
- task_manager.py

6.6 task_manager.FollowFoodTrail Class Reference

Inheritance diagram for task_manager.FollowFoodTrail:



Collaboration diagram for task_manager.FollowFoodTrail:



Public Member Functions

- def __init__
- def start_task
- def perform_task

Public Attributes

- new_task

6.6.1 Detailed Description

```
docstring for FollowFoodTrail
```

Definition at line 153 of file task_manager.py.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 def task_manager.FollowFoodTrail.__init__ (*self, ant*)

Definition at line 155 of file task_manager.py.

6.6.3 Member Function Documentation

6.6.3.1 def task_manager.FollowFoodTrail.perform_task (*self*)

```
if food is found take food_scent_strength
otherwise rank cells based on scent and follow it
if scent trail is lost, return to explore mode
```

Definition at line 161 of file task_manager.py.

6.6.3.2 def task_manager.FollowFoodTrail.start_task (*self*)

Definition at line 158 of file task_manager.py.

6.6.4 Member Data Documentation

6.6.4.1 task_manager.FollowFoodTrail.new_task

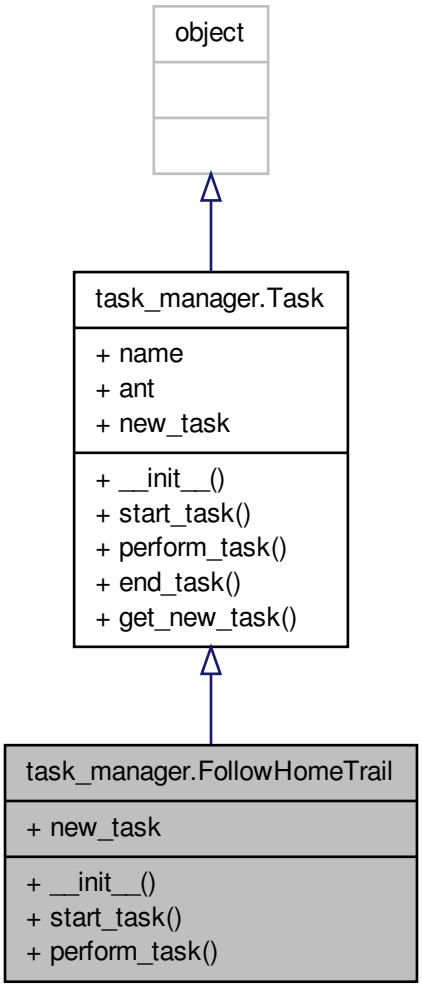
Definition at line 172 of file task_manager.py.

The documentation for this class was generated from the following file:

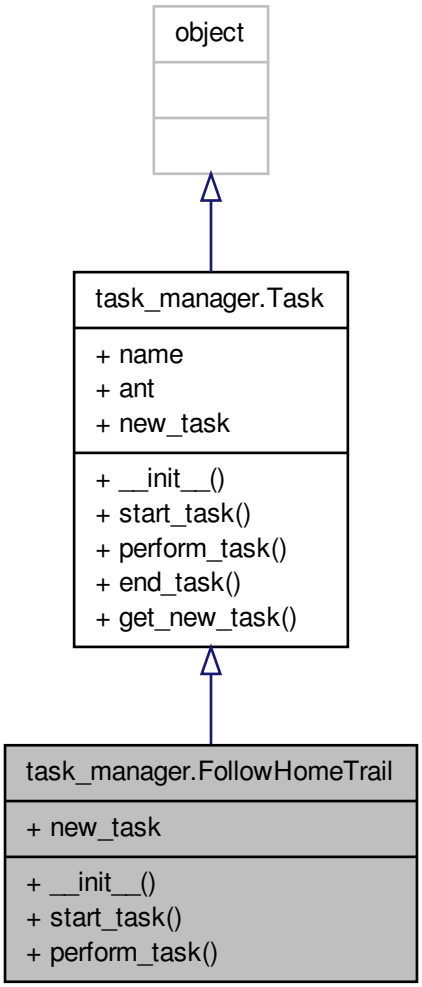
- task_manager.py

6.7 task_manager.FollowHomeTrail Class Reference

Inheritance diagram for task_manager.FollowHomeTrail:



Collaboration diagram for task_manager.FollowHomeTrail:



Public Member Functions

- def __init__
- def start_task
- def perform_task

Public Attributes

- new_task

6.7.1 Detailed Description

```
docstring for FollowFoodTrail
```

Definition at line 183 of file task_manager.py.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 def task_manager.FollowHomeTrail.__init__ (*self*, *ant*)

Definition at line 185 of file task_manager.py.

6.7.3 Member Function Documentation

6.7.3.1 def task_manager.FollowHomeTrail.perform_task (*self*)

```
If home is reached drop the food_scent_strength
If trail is lost return to explore mode
else rank the cell by home scent strength and follow itself
```

Definition at line 191 of file task_manager.py.

6.7.3.2 def task_manager.FollowHomeTrail.start_task (*self*)

Definition at line 188 of file task_manager.py.

6.7.4 Member Data Documentation

6.7.4.1 task_manager.FollowHomeTrail.new_task

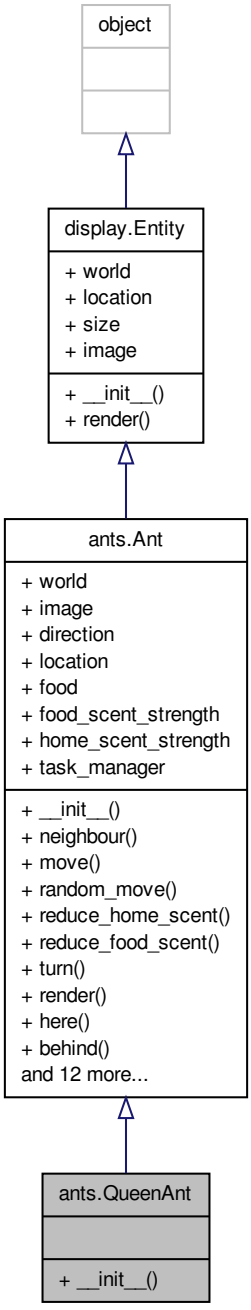
Definition at line 203 of file task_manager.py.

The documentation for this class was generated from the following file:

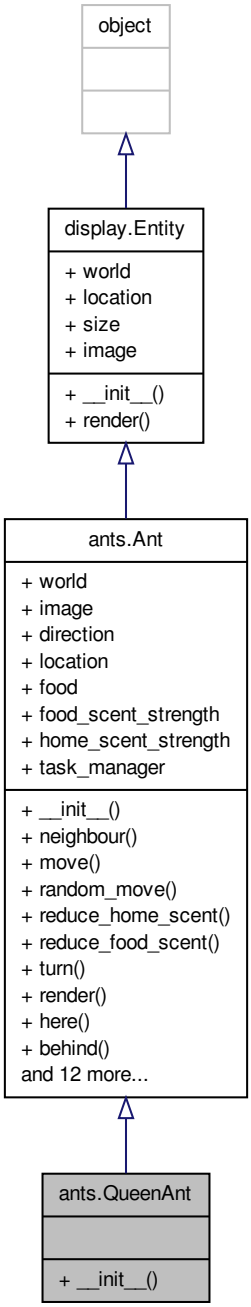
- task_manager.py

6.8 ants.QueenAnt Class Reference

Inheritance diagram for ants.QueenAnt:



Collaboration diagram for ants.QueenAnt:



Public Member Functions

- def __init__

Additional Inherited Members

6.8.1 Detailed Description

```
Ants that produces offsprings and populates the colony
```

Definition at line 223 of file ants.py.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 def ants.QueenAnt.__init__ (*self, world, image, direction, location*)

```
Tasks assigned:
    - HaveFood
    - Produce offsprings
Default task:
    - Have Food
```

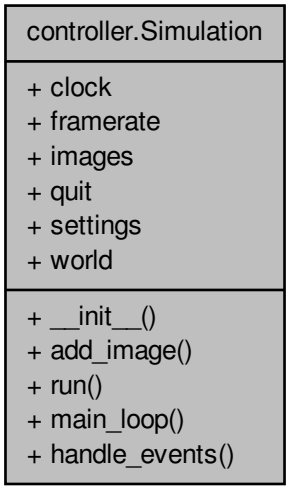
Definition at line 225 of file ants.py.

The documentation for this class was generated from the following file:

- ants.py

6.9 controller.Simulation Class Reference

Collaboration diagram for controller.Simulation:



Public Member Functions

- def __init__
- def add_image
- def run
- def main_loop
- def handle_events

Public Attributes

- clock
- framerate
- images
- quit
- settings
- world

6.9.1 Detailed Description

```
Controls the simulation

    - Runs the main loop
    - Detects mouse, keyboard and other events
    - Loads the necessary images
    - Controls the framerate of the smiulation
```

Definition at line 5 of file controller.py.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 def controller.Simulation.__init__ (*self*)

Definition at line 13 of file controller.py.

6.9.3 Member Function Documentation

6.9.3.1 def controller.Simulation.add_image (*self, name, path*)

```
Loads an image
```

Definition at line 37 of file controller.py.

6.9.3.2 def controller.Simulation.handle_events (*self*)

```
Handles all keyboard, mouse and events like QUIT, etc
```

Definition at line 59 of file controller.py.

6.9.3.3 def controller.Simulation.main_loop (*self*)

```
Updates the simulation
    - Draws the world and update it
    - Handles all user events
    - Controls frame rate
```

Definition at line 46 of file controller.py.

6.9.3.4 def controller.Simulation.run (*self*)

```
Runs the main loop till the user quits
```

Definition at line 41 of file controller.py.

6.9.4 Member Data Documentation

6.9.4.1 controller.Simulation.clock

Definition at line 14 of file controller.py.

6.9.4.2 controller.Simulation.framerate

Definition at line 15 of file controller.py.

6.9.4.3 controller.Simulation.images

Definition at line 16 of file controller.py.

6.9.4.4 controller.Simulation.quit

Definition at line 18 of file controller.py.

6.9.4.5 controller.Simulation.settings

Definition at line 29 of file controller.py.

6.9.4.6 controller.Simulation.world

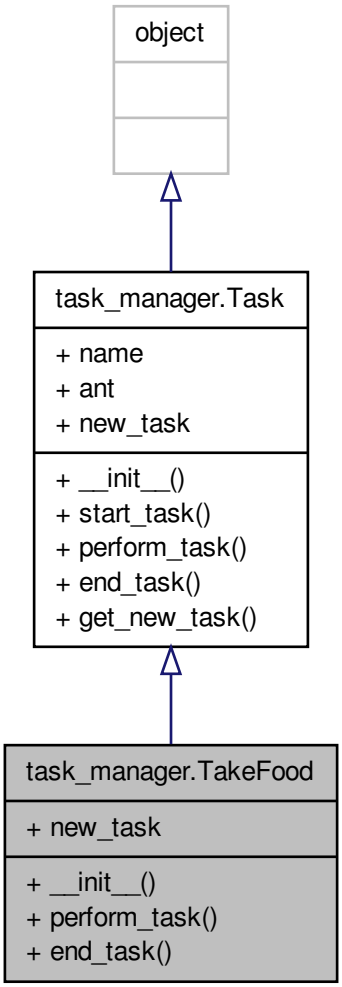
Definition at line 35 of file controller.py.

The documentation for this class was generated from the following file:

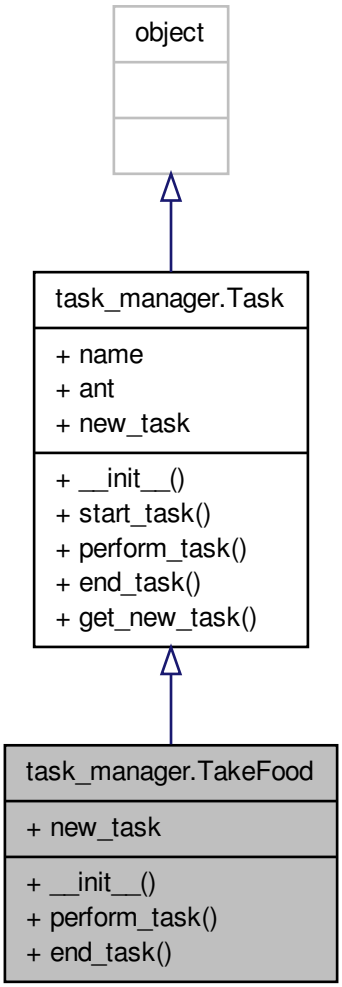
- controller.py

6.10 task_manager.TakeFood Class Reference

Inheritance diagram for task_manager.TakeFood:



Collaboration diagram for task_manager.TakeFood:



Public Member Functions

- def __init__
- def perform_task
- def end_task

Public Attributes

- new_task

6.10.1 Detailed Description

```
Gathering Food Task
```

Definition at line 103 of file task_manager.py.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 def task_manager.TakeFood.__init__ (*self, ant*)

Definition at line 105 of file task_manager.py.

6.10.3 Member Function Documentation

6.10.3.1 def task_manager.TakeFood.end_task (*self*)

```
Increase food_scent_strength
and reduce home_scent_strength
```

Definition at line 119 of file task_manager.py.

6.10.3.2 def task_manager.TakeFood.perform_task (*self*)

```
Take food if available otherwise return to explore mode
```

Definition at line 108 of file task_manager.py.

6.10.4 Member Data Documentation

6.10.4.1 task_manager.TakeFood.new_task

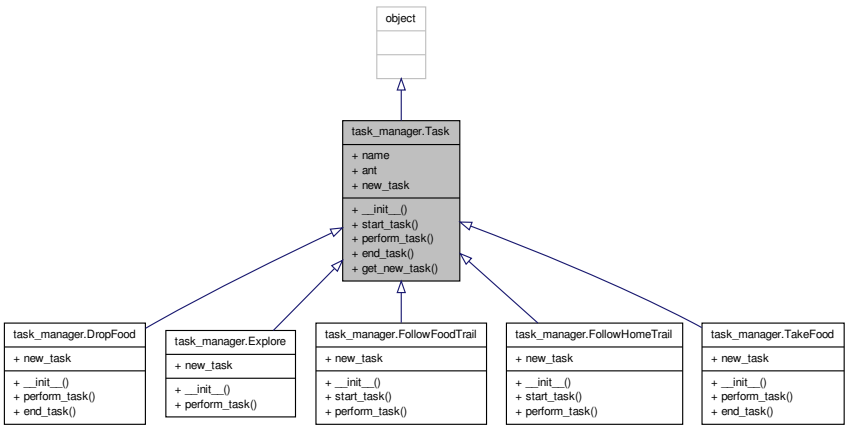
Definition at line 115 of file task_manager.py.

The documentation for this class was generated from the following file:

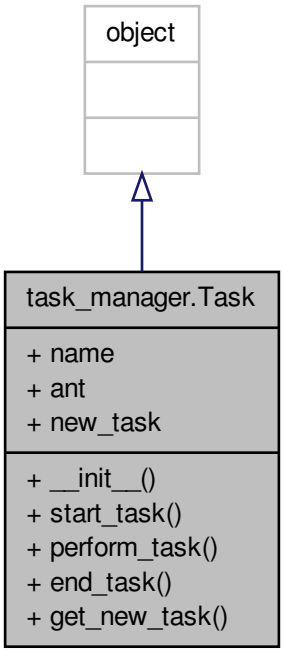
- task_manager.py

6.11 task_manager.Task Class Reference

Inheritance diagram for task_manager.Task:



Collaboration diagram for task_manager.Task:



Public Member Functions

- def __init__
- def start_task
- def perform_task
- def end_task
- def get_new_task

Public Attributes

- name
- ant
- new_task

6.11.1 Detailed Description

```
Base class for a Task
```

Definition at line 31 of file task_manager.py.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 def task_manager.Task.__init__ (*self, name, ant*)

Definition at line 33 of file task_manager.py.

6.11.3 Member Function Documentation

6.11.3.1 def task_manager.Task.end_task (*self*)

```
Actions at the end of a new task
```

Definition at line 46 of file task_manager.py.

6.11.3.2 def task_manager.Task.get_new_task (*self*)

```
Returns and resets new task
```

Definition at line 50 of file task_manager.py.

6.11.3.3 def task_manager.Task.perform_task (*self*)

```
Actions done for a task
```

Definition at line 42 of file task_manager.py.

6.11.3.4 def task_manager.Task.start_task (*self*)

```
Actions at the beginning of a new task
```

Definition at line 38 of file task_manager.py.

6.11.4 Member Data Documentation

6.11.4.1 task_manager.Task.ant

Definition at line 35 of file task_manager.py.

6.11.4.2 task_manager.Task.name

Definition at line 34 of file task_manager.py.

6.11.4.3 task_manager.Task.new_task

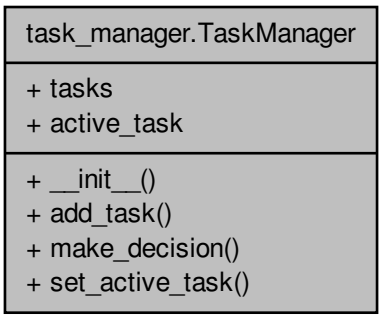
Definition at line 36 of file task_manager.py.

The documentation for this class was generated from the following file:

- task_manager.py

6.12 task_manager.TaskManager Class Reference

Collaboration diagram for task_manager.TaskManager:



Public Member Functions

- def __init__
- def add_task
- def make_decision
- def set_active_task

Public Attributes

- tasks
- active_task

6.12.1 Detailed Description

```
Decides and performs all actions of an Ant
```

Definition at line 3 of file task_manager.py.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 def task_manager.TaskManager.__init__ (*self*)

Definition at line 5 of file task_manager.py.

6.12.3 Member Function Documentation

6.12.3.1 def task_manager.TaskManager.add_task (*self, task*)

```
Adds a new task
```

Definition at line 9 of file task_manager.py.

6.12.3.2 def task_manager.TaskManager.make_decision (*self*)

```
Performs the active task
Checks for new task
If new task is present end the current task and
start the new task
```

Definition at line 13 of file task_manager.py.

6.12.3.3 def task_manager.TaskManager.set_active_task (*self, task_name*)

```
Sets the active task
```

Definition at line 27 of file task_manager.py.

6.12.4 Member Data Documentation

6.12.4.1 task_manager.TaskManager.active_task

Definition at line 7 of file task_manager.py.

6.12.4.2 task_manager.TaskManager.tasks

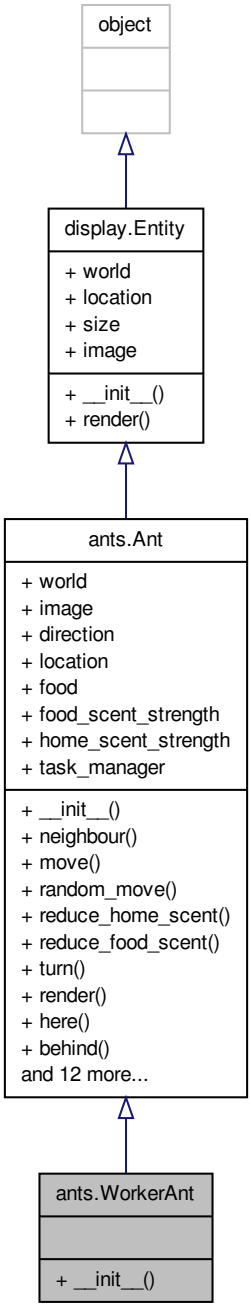
Definition at line 6 of file task_manager.py.

The documentation for this class was generated from the following file:

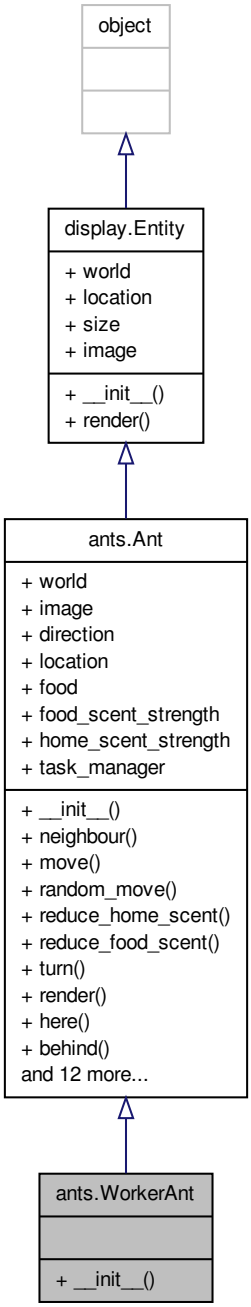
- task_manager.py

6.13 ants.WorkerAnt Class Reference

Inheritance diagram for ants.WorkerAnt:



Collaboration diagram for ants.WorkerAnt:



Public Member Functions

- def __init__

Additional Inherited Members

6.13.1 Detailed Description

```
Ants that explores for foodsource and collects food
```

Definition at line 201 of file ants.py.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 def ants.WorkerAnt.__init__ (*self, world, image, direction, location*)

```
Tasks assigned:
    - Explore
    - TakeFood
    - DropFood
    - FollowFoodTrail
    - FollowHomeTrail
Default task:
    - Explore
```

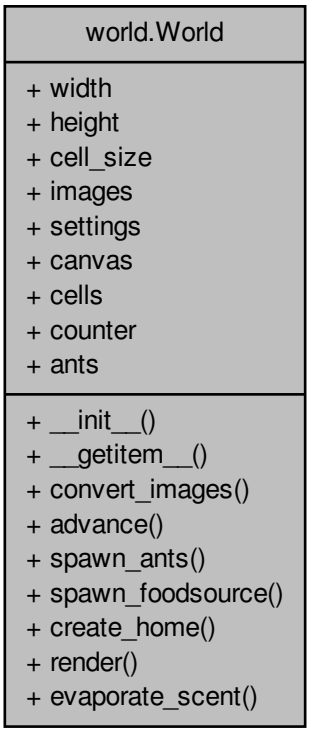
Definition at line 203 of file ants.py.

The documentation for this class was generated from the following file:

- ants.py

6.14 world.World Class Reference

Collaboration diagram for world.World:



Public Member Functions

- def __init__
- def __getitem__
- def convert_images
- def advance
- def spawn_ants
- def spawn_foodsource
- def create_home
- def render
- def evaporate_scent

Public Attributes

- width
- height
- cell_size
- images
- settings
- canvas

- cells
- counter
- ants

6.14.1 Detailed Description

```
Encapsulation of all objects in the simulation
```

Definition at line 96 of file world.py.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 def world.World.__init__ (*self, width, height, cell_size, images, settings*)

```
- Initialise the screen
- Fill screen with "Cells"
- Convert images to pygame format
- Spawn ants, food sources, obstacles, ant home, etc
```

Definition at line 98 of file world.py.

6.14.3 Member Function Documentation

6.14.3.1 def world.World.__getitem__ (*self, location*)

```
Returns the cell at the location
```

Definition at line 124 of file world.py.

6.14.3.2 def world.World.advance (*self*)

```
Advance the simulation by one step
    - Update te ants
    - Evaporate all scents
```

Definition at line 134 of file world.py.

6.14.3.3 def world.World.convert_images (*self*)

```
Convert images to pygame optimised format
```

Definition at line 129 of file world.py.

6.14.3.4 def world.World.create_home (*self*)

```
Create a nest for ants
```

Definition at line 160 of file world.py.

6.14.3.5 def world.World.evaporate_scent (*self*)

```
Evaporates all scent ( uses decay law ) at a rate defined in settings
```

Definition at line 181 of file world.py.

6.14.3.6 def world.World.render (*self*)

```
Draws the world on the screen
```

Definition at line 168 of file world.py.

6.14.3.7 def world.World.spawn_ants (*self*)

```
Spawns ants
```

Definition at line 144 of file world.py.

6.14.3.8 def world.World.spawn_foodsource (*self*)

```
Spawns food sources
```

Definition at line 152 of file world.py.

6.14.4 Member Data Documentation

6.14.4.1 world.World.ants

Definition at line 118 of file world.py.

6.14.4.2 world.World.canvas

Definition at line 111 of file world.py.

6.14.4.3 world.World.cell_size

Definition at line 107 of file world.py.

6.14.4.4 world.World.cells

Definition at line 114 of file world.py.

6.14.4.5 world.World.counter

Definition at line 116 of file world.py.

6.14.4.6 world.World.height

Definition at line 106 of file world.py.

6.14.4.7 world.World.images

Definition at line 108 of file world.py.

6.14.4.8 world.World.settings

Definition at line 109 of file world.py.

6.14.4.9 world.World.width

Definition at line 105 of file world.py.

The documentation for this class was generated from the following file:

- world.py

Chapter 7

File Documentation

7.1 ants.py File Reference

Classes

- class ants.Ant
- class ants.WorkerAnt
- class ants.QueenAnt

Namespaces

- ants

7.2 ants.py

```
00001 from constants import DIRECTIONS
00002 from task_manager import TaskManager, Explore, TakeFood, FollowHomeTrail, FollowFoodTrail, DropFood
00003 from display import Entity
00004 from random import choice, randint
00005
00006 class Ant(Entity):
00007     """A virtual base class for Ants"""
00008     def __init__(self, world, image, direction, location):
00009         super(Ant, self).__init__(world, location, [world.cell_size]*2, image)
00010         self.world = world
00011         self.image = image
00012         self.direction = direction
00013         self.location = location
00014         self.food = 0
00015         self.food_scent_strength = 0
00016         self.home_scent_strength = 0
00017
00018         self.task_manager = TaskManager()
00019
00020     def neighbour(self, direction):
00021         """Returns location of neighbouring cell in a direction
00022         relative to the ant direction"""
00023         x, y = self.location
00024         dx, dy = DIRECTIONS[(self.direction + direction)%8]
00025         return (x+dx)%self.world.width, (y+dy)%self.world.height
00026
00027     def move(self):
00028         """Move the ant by a unit,
00029         Leave a scent trail,
00030         remove the ant from its old cell, and
00031         update the current cell ant with itself
00032         """
00033         new_location = self.neighbour(0)
00034         self.location = new_location
00035         self.behind().ant = None
00036         if not self.behind().is_obstacle():
00037             self.behind().add_home_scent(self.home_scent_strength)
```

```
00038             self.behind().add_food_scent(self.food_scent_strength)
00039         self.here().ant = self
00040
00041     def random_move(self):
00042         """Ant makes a move forward or turns randomly"""
00043         if randint(1,8) == 1:
00044             self.turn( choice([-1, 1]) )
00045         else:
00046             self.move()
00047
00048     def reduce_home_scent(self, amt=1):
00049         """Reduce home scent by 'amt'"""
00050         self.home_scent_strength = max(0, self.
      home_scent_strength-amt)
00051         return self
00052
00053     def reduce_food_scent(self, amt=1):
00054         """Reduce food scent by 'amt'"""
00055         self.food_scent_strength = max(0, self.
      food_scent_strength-amt)
00056         return self
00057
00058     def turn(self, n):
00059         """Changes direction n times"""
00060         self.direction = (self.direction + n) % 8
00061
00062     def render(self):
00063         """Render itself"""
00064         if self.has_food():
00065             super(Ant, self).render(8)
00066         else:
00067             super(Ant, self).render(self.direction)
00068
00069     def here(self):
00070         """The cell it is standing on"""
00071         return self.world[self.location]
00072
00073     def behind(self):
00074         """The cell just behind"""
00075         return self.world[self.neighbour(4)]
00076
00077     def ahead(self):
00078         """The cell just ahead"""
00079         return self.world[self.neighbour(0)]
00080
00081     def ahead_left(self):
00082         """The cell just ahead-left"""
00083         return self.world[self.neighbour(-1)]
00084
00085     def ahead_right(self):
00086         """The cell just ahead-right"""
00087         return self.world[self.neighbour(1)]
00088
00089     def locate_food_nearby(self):
00090         """Locate all sources nearby and return any one randomly
00091         return None if no food source is found"""
00092         directions = []
00093         if self.ahead().has_food():
00094             directions.append(0)
00095         else:
00096             for i in xrange(1, 8):
00097                 if self.world[self.neighbour(i)].has_food():
00098                     directions.append(i)
00099
00100         if directions:
00101             return choice(directions)
00102         else:
00103             return None
00104
00105     def locate_home_nearby(self):
00106         """Locate home cell nearby and return any one randomly
00107         return None if not found"""
00108         directions = []
00109         if self.ahead().is_food():
00110             directions.append(0)
00111         else:
00112             for i in xrange(1, 8):
00113                 if self.world[self.neighbour(i)].is_home():
00114                     directions.append(i)
00115
00116         if directions:
00117             return choice(directions)
00118         else:
00119             return None
00120
00121     def locate_home_scent_nearby(self):
00122         """Scan the 5 directions near the direction of the ant for home scent and
```

```
00123         return one random direction
00124         return None if not found
00125         """
00126         directions = []
00127         if self.ahead().home_scent > 0:
00128             directions.append(0)
00129         else:
00130             for i in xrange(-2, 3):
00131                 if self.world[self.neighbour(i)].home_scent > 0:
00132                     for x in xrange(1,11-5*abs(i)):
00133                         directions.append(i)
00134 
00135         if directions:
00136             return choice(directions)
00137         else:
00138             return None
00139 
00140     def locate_food_scent_nearby(self):
00141         """Scan the 5 directions near the direction of the ant for food scent and
00142         return one random direction
00143         return None if not found
00144         """
00145         directions = []
00146         if self.ahead().food_scent > 0:
00147             directions.append(0)
00148         else:
00149             for i in xrange(-2, 3):
00150                 if self.world[self.neighbour(i)].food_scent > 0:
00151                     for x in xrange(1,11-5*abs(i)):
00152                         directions.append(i)
00153 
00154         if directions:
00155             return choice(directions)
00156         else:
00157             return None
00158 
00159     def rank_by_food_scent(self):
00160         """Scan the 5 directions near the direction of the ant for food scent and
00161         return the direction with the strongest scent
00162         return None if not found
00163         """
00164         best_direction = 0
00165         best_direction_scent = 0
00166         for i in [0, -1, 1, -1, 2]:
00167             cell = self.world[self.neighbour(i)]
00168             if cell.food_scent > best_direction_scent:
00169                 best_direction = i
00170                 best_direction_scent = cell.food_scent
00171         return best_direction
00172 
00173     def rank_by_home_scent(self):
00174         """Scan the 5 directions near the direction of the ant for home scent and
00175         return the direction with the strongest scent
00176         return None if not found
00177         """
00178         best_direction = 0
00179         best_direction_scent = 0
00180         for i in [0, -1, 1, -1, 2]:
00181             cell = self.world[self.neighbour(i)]
00182             if cell.home_scent > best_direction_scent:
00183                 best_direction = i
00184                 best_direction_scent = cell.home_scent
00185         return best_direction
00186 
00187     def drop_food(self):
00188         """
00189         Set food to zero
00190         Update the food values of the home cell it reached
00191         """
00192         self.food = 0
00193 
00194     def has_food(self):
00195         return bool(self.food)
00196 
00197     def __nonzero__(self):
00198         return True
00199 
00200 
00201 class WorkerAnt(Ant):
00202     """Ants that explores for foodsource and collects food"""
00203     def __init__(self, world, image, direction, location):
00204         """
00205         Tasks assigned:
00206             - Explore
00207             - TakeFood
00208             - DropFood
00209             - FollowFoodTrail
```

```
00210             - FollowHomeTrail
00211         Default task:
00212             - Explore
00213         """
00214         Ant.__init__(self, world, image, direction, location)
00215         self.task_manager.add_task(Explore(self))
00216         self.task_manager.add_task(TakeFood(self))
00217         self.task_manager.add_task(DropFood(self))
00218         self.task_manager.add_task(FollowFoodTrail(self))
00219         self.task_manager.add_task(FollowHomeTrail(self))
00220         self.task_manager.set_active_task("explore")
00221
00222
00223 class QueenAnt(Ant):
00224     """Ants that produces offsprings and populates the colony"""
00225     def __init__(self, world, image, direction, location):
00226         """
00227         Tasks assigned:
00228             - HaveFood
00229             - Produce offsprings
00230         Default task:
00231             - Have Food
00232         """
00233         Ant.__init__(self, world, image, direction, location)
00234         raise NotImplementedError
```

7.3 constants.py File Reference

Namespaces

- constants

Variables

- tuple constants.DIRECTIONS
- int constants.WIDTH = 80
- int constants.HEIGHT = 60

7.4 constants.py

```
00001 DIRECTIONS = ((1,0), (1,1), (0,1), (-1,1),
00002     (-1,0), (-1,-1), (0,-1), (1,-1))
00003 WIDTH = 80
00004 HEIGHT = 60
00005 WHITE = (255,255,255)
```

7.5 controller.py File Reference

Classes

- class controller.Simulation

Namespaces

- controller

7.6 controller.py

```
00001 from pygame import image, time, key, event
00002 from pygame.constants import *
```

```
00003 from world import World
00004
00005 class Simulation():
00006     """Controls the simulation
00007
00008         - Runs the main loop
00009         - Detects mouse, keyboard and other events
00010         - Loads the necessary images
00011         - Controls the framerate of the smiulation
00012     """
00013     def __init__(self):
00014         self.clock = time.Clock()
00015         self.framerate = 60
00016         self.images = {}
00017
00018         self.quit = False
00019
00020         self.add_image("ant", "ant.png")
00021         self.add_image("grass", "grass.png")
00022         self.add_image("food", "food.png")
00023         self.add_image("home", "home.png")
00024         self.add_image("obstacle", "obstacle.png")
00025         self.add_image("home_scent", "home_scent.png")
00026         self.add_image("food_scent", "food_scent.png")
00027         self.add_image("cell", "cell.png")
00028
00029         self.settings = {
00030         "no_of_ants": 50,
00031         "evaporation_rate": .95,
00032         "home_size": 10
00033         }
00034
00035         self.world = World(80, 60, 10, self.images, self.
    settings)
00036
00037     def add_image(self, name, path):
00038         """ Loads an image"""
00039         self.images[name] = image.load('images/' + path)
00040
00041     def run(self):
00042         """Runs the main loop till the user quits"""
00043         while self.quit is False:
00044             self.main_loop()
00045
00046     def main_loop(self):
00047         """Updates the simulation
00048             - Draws the world and update it
00049             - Handles all user events
00050             - Controls frame rate
00051         """
00052         self.world.render()
00053         self.world.advance()
00054
00055         self.handle_events()
00056
00057         self.clock.tick(self.framerate)
00058
00059     def handle_events(self):
00060         """Handles all keyboard, mouse and events like QUIT, etc"""
00061         keys = key.get_pressed()
00062         if keys[K_q] or keys[K_ESCAPE]:
00063             self.quit = True
00064         for evt in event.get():
00065             if evt.type == QUIT:
00066                 self.quit = True
```

7.7 display.py File Reference

Classes

- class display.Entity

Namespaces

- display

7.8 display.py

```
00001 class Entity(object):
00002     """Base class for all drawable objects"""
00003     def __init__(self, world, location, size, image):
00004         self.world = world
00005         self.location = location
00006         self.size = size
00007         self.image = image
00008 
00009     def render(self, index=0):
00010         """Draw the object into the screen
00011             - selects the portion of the image to draw from the "index" argument
00012         """
00013         x, y = self.location
00014         x *= self.world.cell_size
00015         y *= self.world.cell_size
00016         w, h = self.size
00017         position = (x, y)
00018         patch_rect = (w*index, 0, w, h)
00019         self.world.canvas.blit(self.image, position, patch_rect)
```

7.9 main.py File Reference

Namespaces

- main

Variables

- tuple main.simulation = Simulation()

7.10 main.py

```
00001 from controller import Simulation
00002 
00003 if __name__ == '__main__':
00004     simulation = Simulation()
00005     simulation.run()
```

7.11 task_manager.py File Reference

Classes

- class task_manager.TaskManager
- class task_manager.Task
- class task_manager.Explore
- class task_manager.TakeFood
- class task_manager.DropFood
- class task_manager.FollowFoodTrail
- class task_manager.FollowHomeTrail

Namespaces

- task_manager

7.12 task_manager.py

```
00001 from random import randint, choice
00002
00003 class TaskManager():
00004     """Decides and performs all actions of an Ant"""
00005     def __init__(self):
00006         self.tasks = {}
00007         self.active_task = None
00008
00009     def add_task(self, task):
00010         """Adds a new task"""
00011         self.tasks[task.name] = task
00012
00013     def make_decision(self):
00014         """Performs the active task
00015         Checks for new task
00016         If new task is present end the current task and
00017         start the new task"""
00018         self.active_task.perform_task()
00019
00020         new_task = self.active_task.get_new_task()
00021
00022         if new_task:
00023             self.active_task.end_task()
00024             self.set_active_task(new_task)
00025             self.active_task.start_task()
00026
00027     def set_active_task(self, task_name):
00028         """Sets the active task"""
00029         self.active_task = self.tasks[task_name]
00030
00031 class Task(object):
00032     """Base class for a Task"""
00033     def __init__(self, name, ant):
00034         self.name = name
00035         self.ant = ant
00036         self.new_task = None
00037
00038     def start_task(self):
00039         """Actions at the beginning of a new task"""
00040         pass
00041
00042     def perform_task(self):
00043         """Actions done for a task"""
00044         pass
00045
00046     def end_task(self):
00047         """Actions at the end of a new task"""
00048         pass
00049
00050     def get_new_task(self):
00051         """Returns and resets new task"""
00052         new_task = self.new_task
00053         self.new_task = None
00054         return new_task
00055
00056 class Explore(Task):
00057     """Ant Exploring Task"""
00058     def __init__(self, ant):
00059         super(Explore, self).__init__("explore", ant)
00060
00061     def perform_task(self):
00062         """
00063          If ant has food -
00064              find home nearby and drop food there, else
00065              find home scent nearby and switch to that task, else
00066              make a random move
00067          If ant is searching for food
00068              if food is found switch to take food task, else
00069              find a food scent trail, else
00070              avoid obstacles
00071              reverse direction if it finds home nearby
00072          Reduce it scent strength by an unit
00073         """
00074         ant = self.ant
00075         if ant.has_food():
00076             home_nearby = ant.locate_home_nearby()
00077             home_scent_nearby = ant.locate_home_scent_nearby()
00078             if home_nearby != None:
00079                 self.new_task = "drop food"
00080             elif home_scent_nearby != None:
00081                 ant.turn(home_scent_nearby)
00082                 self.new_task = "follow home trail"
00083             else:
00084                 ant.random_move()
```

```
00085         else:
00086             food_nearby = ant.locate_food_nearby()
00087             food_scent_nearby = ant.locate_food_scent_nearby()
00088             if food_nearby != None:
00089                 ant.turn(food_nearby)
00090                 self.new_task = "take food"
00091             elif ant.ahead().is_obstacle() or ant.ahead().has_ant():
00092                 ant.turn(randint(1,3)-2)
00093             elif ant.ahead().is_home():
00094                 ant.turn(4)
00095                 ant.home_scent_strength = 40
00096             elif ant.ahead().food_scent > 1 and ant.food <= 0:
00097                 ant.move()
00098                 self.new_task = "follow food trail"
00099             else:
00100                 ant.random_move()
00101         ant.reduce_food_scent(1).reduce_home_scent(1)
00102
00103 class TakeFood(Task):
00104     """Gathering Food Task"""
00105     def __init__(self, ant):
00106         super(TakeFood, self).__init__("take food", ant)
00107
00108     def perform_task(self):
00109         """Take food if available otherwise return to explore mode"""
00110         ant = self.ant
00111         food = ant.ahead().get_food(1)
00112         if food:
00113             ant.food = food
00114             ant.turn(4)
00115             self.new_task = "follow home trail"
00116         else:
00117             self.new_task = "explore"
00118
00119     def end_task(self):
00120         """Increase food_scent_strength
00121         and reduce home_scent_strength"""
00122         self.ant.food_scent_strength = 40
00123         self.ant.home_scent_strength = 0
00124
00125
00126 class DropFood(Task):
00127     """Drop Food Task"""
00128     def __init__(self, ant):
00129         super(DropFood, self).__init__("drop food", ant)
00130
00131     def perform_task(self):
00132         """
00133         If ant reaches home drop the food inside the home
00134         otherwise follow a home trail
00135         """
00136         ant = self.ant
00137         home_nearby = ant.locate_home_nearby()
00138         if home_nearby !=None:
00139             ant.turn(home_nearby)
00140             ant.drop_food()
00141             ant.turn(4)
00142             self.new_task = "explore"
00143         else:
00144             self.new_task = "follow home trail"
00145
00146     def end_task(self):
00147         """
00148         Increase home scent strength and reduce food scent strength
00149         """
00150         self.ant.food_scent_strength = 0
00151         self.ant.home_scent_strength = 40
00152
00153 class FollowFoodTrail(Task):
00154     """docstring for FollowFoodTrail"""
00155     def __init__(self, ant):
00156         super(FollowFoodTrail, self).__init__("follow food trail", ant)
00157
00158     def start_task(self):
00159         pass
00160
00161     def perform_task(self):
00162         """
00163         if food is found take food_scent_strength
00164         otherwise rank cells based on scent and follow it
00165         if scent trail is lost, return to explore mode
00166         """
00167         ant = self.ant
00168         food_nearby = ant.locate_food_nearby()
00169         food_scent_nearby = ant.locate_food_scent_nearby()
00170         if food_nearby != None:
00171             ant.turn(food_nearby)
```

```
00172             self.new_task = "take food"
00173         elif ant.ahead().is_obstacle() or ant.ahead().has_ant():
00174             ant.turn(randint(1,3)-2)
00175         elif ant.ahead().is_home():
00176             ant.turn(4)
00177             ant.home_scent_strength = 40
00178         else:
00179             ant.turn(ant.rank_by_food_scent())
00180             ant.move()
00181         ant.reduce_home_scent(1).reduce_food_scent(1)
00182
00183 class FollowHomeTrail(Task):
00184     """docstring for FollowFoodTrail"""
00185     def __init__(self, ant):
00186         super(FollowHomeTrail, self).__init__("follow home trail", ant)
00187
00188     def start_task(self):
00189         pass
00190
00191     def perform_task(self):
00192         """
00193             If home is reached drop the food_scent_strength
00194             If trail is lost return to explore mode
00195             else rank the cell by home scent strength and follow itself
00196         """
00197         ant = self.ant
00198         home_nearby = ant.locate_home_nearby()
00199         if ant.ahead().is_obstacle() or ant.ahead().has_ant() or ant.ahead().is_food():
00200             ant.turn(choice([-1, 1]))
00201         elif home_nearby != None:
00202             ant.turn(home_nearby)
00203             self.new_task = "drop food"
00204         else:
00205             ant.turn(ant.rank_by_home_scent())
00206             ant.move()
00207         ant.reduce_food_scent(1).reduce_home_scent(1)
```

7.13 world.py File Reference

Classes

- class world.Cell
- class world.World

Namespaces

- world

7.14 world.py

```
00001 from ants import WorkerAnt
00002 from random import randint, choice
00003 from pygame import display
00004 from display import Entity
00005 from constants import WHITE
00006 from math import sqrt
00007
00008 class Cell(Entity):
00009     """Data containers for each location in World"""
00010     def __init__(self, world, i, j, cell_size):
00011         self.obstacle = False
00012         self.food = 0
00013         self.home_scent = 0
00014         self.food_scent = 0
00015         self.ant = None
00016         self.home = False
00017         super(Cell, self).__init__(world, (i, j), [cell_size]*2, world.images["cell"])
00018
00019     def add_food(self, amt):
00020         self.food += amt
00021
00022     def add_home_scent(self, amt):
00023         self.home_scent += amt
00024
```

```
00025     def add_food_scent(self, amt):
00026         self.food_scent += amt
00027 
00028     def get_food(self, amt):
00029         """Get "amt" amount of food if available else returns whatever food is available"""
00030         if self.food < amt:
00031             food = self.food
00032             self.food = 0
00033             return food
00034         else:
00035             self.food -= amt
00036             return amt
00037 
00038     def is_obstacle(self):
00039         return bool(self.obstacle)
00040 
00041     def is_home(self):
00042         return bool(self.home)
00043 
00044     def is_food(self):
00045         return bool(self.food)
00046 
00047     def has_ant(self):
00048         return bool(self.ant)
00049 
00050     def has_food(self):
00051         return True if self.food > 0 else False
00052 
00053     def make_home(self):
00054         self.home = True
00055 
00056     def make_obstacle(self):
00057         self.obstacle = True
00058 
00059     def evaporate_scent(self, rate):
00060         """Evaporates scent ( decay law )"""
00061         self.food_scent *= rate
00062         self.home_scent *= rate
00063         if self.food_scent < .3:
00064             self.food_scent = 0
00065         if self.home_scent < .3:
00066             self.home_scent = 0
00067 
00068     def render(self):
00069         """Changes "index" to render the cell according to what it represents
00070         (home, food, etc) and calls the super class
00071         Also renders scent levels with transparency depending on its strength
00072         """
00073         if self.is_food():
00074             index = 3
00075         elif self.is_obstacle():
00076             index = 2
00077         elif self.is_home():
00078             index = 1
00079         else:
00080             index = 0
00081 
00082         self.image.set_alpha(255)
00083 
00084         super(Cell, self).render(index)
00085 
00086         if self.home_scent > 0:
00087             index = 4
00088             self.image.set_alpha(self.home_scent)
00089         super(Cell, self).render(index)
00090 
00091         if self.food_scent > 0:
00092             index = 5
00093             self.image.set_alpha(self.food_scent)
00094         super(Cell, self).render(index)
00095 
00096 class World():
00097     """Encapsulation of all objects in the simulation"""
00098     def __init__(self, width, height, cell_size, images, settings):
00099         """
00100         - Initialise the screen
00101         - Fill screen with "Cells"
00102         - Convert images to pygame format
00103         - Spawn ants, food sources, obstacles, ant home, etc
00104         """
00105         self.width = width
00106         self.height = height
00107         self.cell_size = cell_size
00108         self.images = images
00109         self.settings = settings
00110 
00111         self.canvas = display.set_mode((self.width*self.cell_size, self.
```

```
      height*self.cell_size))
00112         self.convert_images()
00113
00114         self.cells = [[Cell(self, i, j, self.cell_size) for j in xrange(height)] for i in
       xrange(width)]
00115
00116         self.counter = 0
00117
00118         self.ants = {}
00119         self.spawn_ants()
00120         for i in xrange(1):
00121             self.spawn_foodsource()
00122         self.create_home()
00123
00124     def __getitem__(self, location):
00125         """Returns the cell at the location"""
00126         x, y = location
00127         return self.cells[x][y]
00128
00129     def convert_images(self):
00130         """Convert images to pygame optimised format"""
00131         for name in self.images:
00132             self.images[name] = self.images[name].convert()
00133
00134     def advance(self):
00135         """Advance the simulation by one step
00136             - Update te ants
00137             - Evaporate all scents
00138         """
00139         for i in self.ants:
00140             self.ants[i].task_manager.make_decision()
00141
00142         self.evaporate_scent()
00143
00144     def spawn_ants(self):
00145         """Spawns ants"""
00146         for i in xrange(self.settings["no_of_ants"]):
00147             direction = randint(0,7)
00148             location = [randint(1,self.width), randint(1,self.height)]
00149             location = [self.width/2-10, self.height/2-10]
00150             self.ants[i] = WorkerAnt(self, self.images["ant"], direction, location)
00151
00152     def spawn_foodsource(self):
00153         """Spawns food sources"""
00154         x, y = randint(0,self.width-1), randint(0,self.height-1)
00155         for i in xrange(randint(2000, 5000)):
00156             dx = randint(-3,3)
00157             dy = choice([-1,1])*randint(0, int(sqrt(9-dx**2)))
00158             self.cells[(x+dx)%self.width][(y+dy)%self.height].add_food(1)
00159
00160     def create_home(self):
00161         """Create a nest for ants"""
00162         n = self.settings["home_size"]
00163         x, y = self.width/2 -5, self.height/2 -5
00164         for i in xrange(n):
00165             for j in xrange(n):
00166                 self.cells[x+i-1][y+j-1].make_home()
00167
00168     def render(self):
00169         """Draws the world on the screen"""
00170         self.canvas.fill(WHITE)
00171
00172         for cells in self.cells:
00173             for cell in cells:
00174                 cell.render()
00175
00176         for ant in self.ants.values():
00177             ant.render()
00178
00179         display.update((0, 0), (self.width*self.cell_size, self.
      height*self.cell_size))
00180
00181     def evaporate_scent(self):
00182         """Evaporates all scent ( uses decay law ) at a rate defined in settings"""
00183         for cells in self.cells:
00184             for cell in cells:
00185                 cell.evaporate_scent(self.settings["evaporation_rate"])
```

Index