CS 1803

Individual Homework 11 – SimWorld

Due: Wednesday, November 24th, before 6 PM
Out of 100 points

Files to submit: **1. HW11.py**

This is an INDIVIDUAL assignment!

Students may only collaborate with fellow students currently taking CS 1803, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

Notes:

- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- **Do not wait until the last minute** to do this assignment in case you run into problems.

Instructions:

In this assignment, you will write a simple predator/prey simulation by sub-classing provided SimulationWorld and Agent classes. Download the SimWorld.py file. If you run this file, it will run a simple simulation with yellow and orange agents. You should NOT edit this file. (The homework code you turn in will NOT contain this file. Instead, your homework will have a statement at the top "from SimWorld import *". When grading your assignment, your TA's will use the official (unchanged) version of this file. Any changes you may have made to your copy of the file will not be considered when grading your homework, so you need to be sure that your HW11.py file works with the provided SimWorld.py file.

When you are creating your own Agents, any behavior or attributes that is not specified in this document should be inherited from the provided superclass.

Green Agent

Create a subclass of the Agent (provided in the SimWorld.py file) called GreenAgent. A GreenAgent is colored green. When a GreenAgent is created, it should move towards the coordinate (0,0). If it reaches (0,0) it should then head for (500,500). If it reaches (500, 500), it should again head towards (0,0), and so on....

Remember that an Agent's action method calls the moveRandomly method. Because you do not want to move randomly, a GreenAgent's action method should NOT call moveRandomly, and instead either call a move method that you write, or do the movement on it's own.

You do NOT have to be super fancy with your movements. Simply look at the X value of yourself, and if it's bigger than the coordinate you are heading for, reduce it by the speed variable inherited from Agent (If it's smaller than the coordinate you are heading for, increase it by the speed variable.) Do the same thing with the Y variable. If you "overshoot" (0,0) or (500,500), just turn around and go for the other point. You don't have to hit them exactly.

Note: When you update your own x and y position, you also need to move the icon by a similar amount. The moveIcon(deltaX, deltaY) function takes in two parameters which are deltas, not absolute parameters. If you moveIcon(5,5) your icon will move down five and over to the right five. If you moveIcon(-5,-5) your icon will move up five and over to the left five.

You can earn 5 EC points for calculating the exact slope that you should head to your destination point, and using it to calculate how much to move in the X and Y dimension. (But don't move in the X or Y dimension faster than the speed variable inherited from Agent!) This will keep your Green Agents from hitting the sides of the world and moving along them until they hit the 0,0 point. It should also keep you from "overshooting" the exact coordinate you are aiming for.

Black Agent

Create a subclass of the Agent called BlackAgent. A BlackAgent is colored black. You will define two methods for your BlackAgent: action(), and eat().

When a BlackAgent's action method is called, it will call the action() method **on it's superclass** (which results in it moving randomly) and then it's own eat() method.

When the eat() method is called, the BlackAgent looks for the closest agent. The easiest way to do this is to use the findClosest method in SimWorld. You are also allowed to iterate through all of the agents stored in the SimWorld yourself, but make sure you do it right!

If the closest agent is a GreenAgent, the BlackAgent will calculate the distance to the GreenAgent. *(The isinstance() built in function is useful to determine if an object is an instance of a particular class. You may find the self.world.distance() function will help with the distance calculation)*. If the GreenAgent is within 15 units of distance, the BlackAgent will "eat" it (Remove it from the simulation.) When a BlackAgent "eats" a green agent, it should print a message to the terminal "Green Agent eaten!".

Red Agent

Create a sublcass of your BlackAgent called RedAgent. A RedAgent is colored red.

Override the action() method so that it implements the following behavior: If the closest agent is a GreenAgent, the RedAgent will move towards it (at the self.speed rate). If the closest agent is NOT a GreenAgent, the RedAgent will move randomly (similar to a generic Agent). After it has moved, it will call the eat() method it has inherited from BlackAgent.

Main Program

Write a main program that instantiates a SimWorld. Then, using the randrange method from the random module, add the following agents at random locations (between 0 and 500 in the X and Y dimension) to your world:

- 10 GreenAgents
- 10 BlackAgents
- 2 Red Agents

Then, begin the simulation and let it run.

Grading

You will earn points as follows for each piece of functionality in your code:

GreenAgent	**25**
-Colored Green	5
-Moves towards (0,0) no matter where it begins	10

-Cycles between (0,0) and (500,500) thereafter	10	
BlackAgent		**30**
-Colored Black	5	
-Inherits Move Randomly behavior from Agent!	5	
-Eats GreenAgents if they come within 15 units	20	
RedAgent		**30**
-Colored Red	5	
-Inherits "eat" behavior from BlackAgent	5	
-If closest agent is a GreenAgent, moves towards it.	15	
-If closest agent is not a GreenAgent, moves randomly.	5	
Main Program		**15**
-Instantiates a SimWorld	2	
-Populates it with 10/10/2 Green/Black/Red Agents	8	
-Runs correctly	5	
Total Possible:		**100**

+5 EC possible for green agents moving directly to (0,0) from any random location (without exceeding the inherited "speed" in either the X or Y dimension)