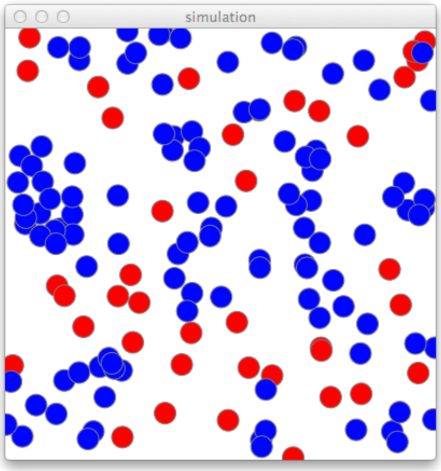
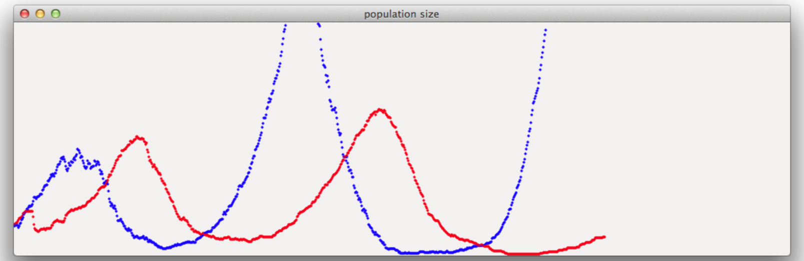
CMSC 143: Object-Oriented Programming with Robots

Lab 12: Predator-Prey Simulation
Due December 1, 2016

Predator-prey simulations are used to understand how populations of animals interact. In this lab's agent-based simulation, there will be two types of animals: Rabbits (prey) and Wolves (predators). Your task is to program the individual animal behaviors. Both types of animals run around randomly and reproduce with some probability. The wolves eat nearby rabbits, removing rabbits from the population. The wolves gain energy from eating the rabbits, but will die if their energy drops to zero. The wolves are born with some energy units and expend energy each timestep. Wolves are less likely to reproduce if their energy is low.

The `Animal` parent class and the `World` class are provided. Your first task is to implement the `Rabbit` and `Wolf` child classes. You **should not** modify the `World` or `Animal` classes. Resist the urge to copy and paste; use inheritance. Next, you should run experiments with your simulation. Gather data about how the populations change as you vary one of the parameters: initial population size, speed, movement strategy, reproduction rate. Use `Window()`[1] to plot the data and include the graph along with a brief reflection in your lab report. The `display` attribute can be used to disable drawing each animal, resulting in faster simulations.





Learning Objectives

◦ Apply Object-Oriented Design ◦ Use Inheritance ◦ Create a Simulation

Deliverables

Submit an electronic copy of your lab using moodle. Your program should have your name, email, assignment description, the date, and collaboration statement at the top of the file as a comment. Your submission should be a zip file that expands to a folder with at least these two files:

```
cmsc143-lab12-LASTNAME-FIRSTNAME
      lab12.py   -- Your program (only adding Wolf and Rabbit)
      lab12.pdf  -- Your simulation results
```

[1]Either using `savePicture(Picture(w), 'results.png')` or use the printscrn button on your keyboard to take a screenshot of your window and then paste that into your report.

```python
from Graphics import *
import random

class World(object):

    def __init__(self, nR, nW):
        ''' Create a simulated world with nR rabbits and nW wolves '''
        self.display = True
        self.nR = nR
        self.nW = nW
        self.animals = []

        self.win = Window("simulation", 400, 400)
        self.win.setBackground(Color("white"))

        self.gwin = Window("population size", 1000, 300)

        for i in range(nR):
            r = Rabbit(self)

        for i in range(nW):
            w = Wolf(self)

    def run(self):
        ''' run the simulated world '''
        t = 0
        while t < self.gwin.getWidth() and self.nR > 0 and self.nW > 0:

            self.nR = 0
            self.nW = 0
            for a in self.animals[:]:
                if a.alive:
                    a.takeAStep()
                    if isinstance(a, Rabbit):
                        self.nR += 1
                    elif isinstance(a, Wolf):
                        self.nW += 1
                else:
                    self.animals.remove(a)

            pr = Circle((t, self.gwin.getHeight() - self.nR), 1)
            pr.color = Color("blue")
            pr.draw(self.gwin)

            pw = Circle((t, self.gwin.getHeight() - self.nW), 1)
            pw.color = Color("red")
            pw.draw(self.gwin)

            t = t + 1

        print ("Simulation Done")

    def addAnimal(self, a):
        self.animals.append(a)
    def nearbyAnimals(self, a):
        ''' find all the nearby animals within sensing range of a'''
        nearby = []
        for o in self.animals:
            if o.alive and a.distance(o) < a.SENSING_RANGE and o != a:
                nearby.append(o)

        return nearby

    def closestNeighbor(self, a, t):
        ''' find the closest animal to animal a of type t;
            return None if no closest exists'''
        closestDistance = 100000000
        neighbor = None
        for o in self.nearbyAnimals(a):
            if isinstance(o, t) and a.distance(o) < closestDistance:
                neighbor = o
                closestDistance = a.distance(o)
        return neighbor

class Animal(object):
    SIZE = 5
    SENSING_RANGE = 30

    def __init__(self, world):
        ''' Create a new Animal'''
        self.world = world
        self.alive = True
        self.vx = 4
        self.vy = 4
        self.reproduction_prob = 0.02
        self.x = random.uniform(0, self.world.win.getWidth())
        self.y = random.uniform(0, self.world.win.getHeight())
        self.size = self.SIZE
        self.world.addAnimal(self)
        self.setAppearance()

    def setAppearance(self):
        self.appearance = Circle(Point(self.x, self.y), self.size)
        if self.world.display:
            self.appearance.draw(self.world.win)

    def eat(self):
        pass

    def reproduce(self):
        pass

    def die(self):
        '''remove this animal from the population'''
        if self.world.display:
            self.appearance.undraw()
        self.alive = False
    def takeAStep(self):
        ''' move the animal for one timestep'''
        dx = random.uniform(-self.vx, self.vx)
        dy = random.uniform(-self.vy, self.vy)
        if self.insideWindow(dx, dy):
            self.x = self.x + dx
            self.y = self.y + dy
            self.appearance.move(dx,dy)

    def distance(self, other):
        ''' find the distance between myself and the other animal'''
        return ((self.x - other.x)**2 + (self.y - other.y)**2)**0.5

    def insideWindow(self, dx, dy):
        ''' check to see if moving animal by (dx, dy) keeps it in the window'''
        tx = self.x + dx
        ty = self.y + dy
        if self.size/2 < tx < self.world.win.getWidth() - self.size/2 and\
           self.size/2 < ty < self.world.win.getHeight() - self.size/2:
            return True
        else:
            return False

    def __str__(self):
        return "Animal at (%d, %d)" % (self.x, self.y)

def go():
    random.seed(5)
    w = World(60, 60)
    w.run()
```