Assignment #5: Critters! 2.0 (Party Edition)

CSCI 364 Fall 2021 Oberlin College

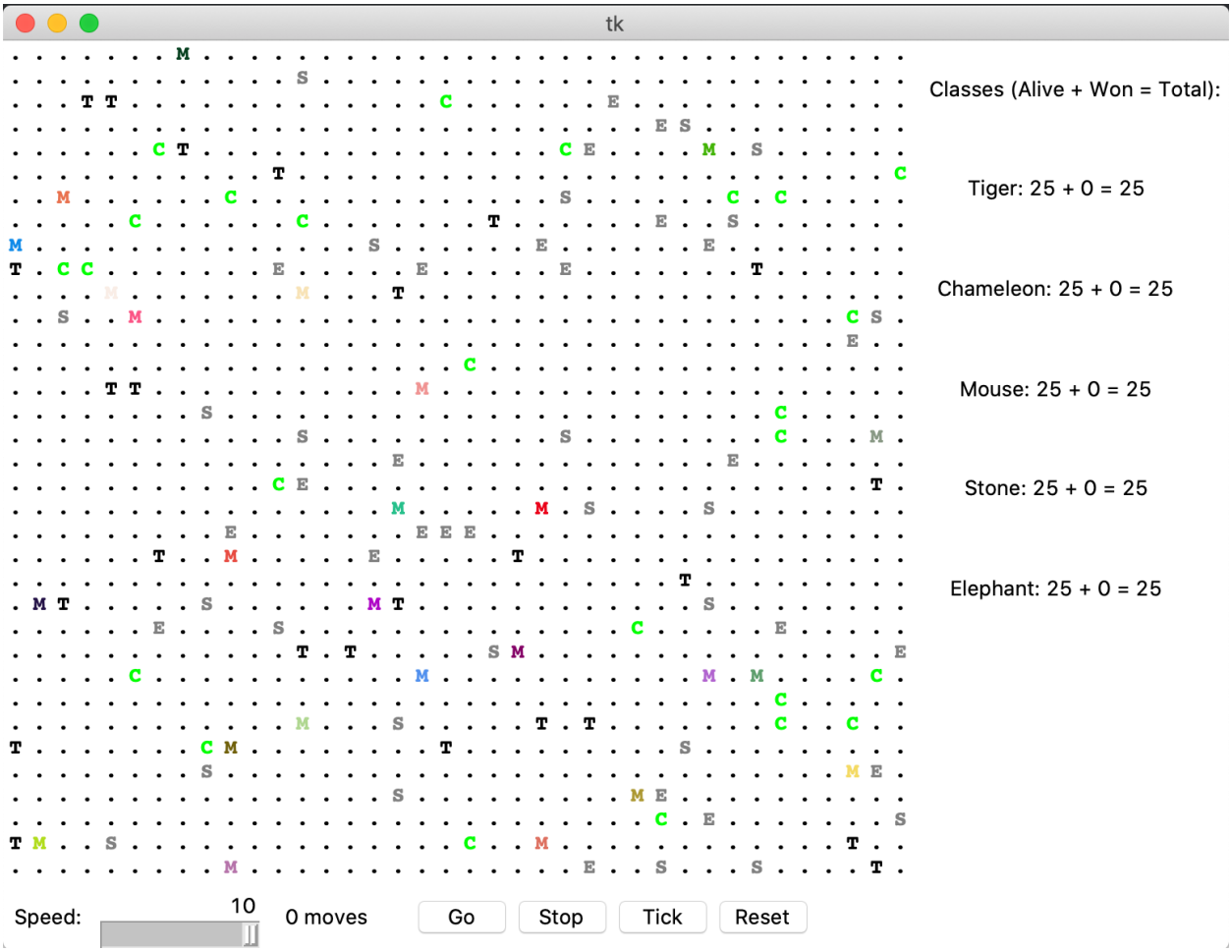
Due: Wednesday January 5 at 11:59 PM

Background

Our final assignment this semester has two goals:

1. Have some fun experimenting with creating intelligent agents that interact in an environment, updating a favorite lab from CSCI 150
2. Compare some existing AI methods with our own creative ideas for AI

In particular, we will be creating Critters that play a modified version of the Critter lab. The original details for the Critter lab can be found at:
https://www.cs.oberlin.edu/~aeck/Spring2021/CSCI150/FinalProject/Critters/index.html



Here, the environment is a rectangular grid where Critters can move around. If two Critters try to occupy the same space on the grid, they will interact with one another. New to this assignment, we have additional types of interactions. Each interaction, Critters choose one of five actions:

1. Three fighting actions (from the original lab): ROAR, SCRATCH, and POUNCE
2. A PARTY action, and
3. A HEAL action

The goal of the Critters is to gain as many **karma** points as they can from their interactions with other Critters.

The rules of interactions are as follows:

1. If both Critters choose to fight, then one is a winner and one loses, based on their chosen actions (similar in structure to Ro-Sham-Bo or Paper-Rock-Scissors): ROAR beats SCRATCH beats POUNCE beats ROAR. The losing Critter loses 25 **health** (starting from a maximum of 100), and the winning Critter gains one karma point. The losing Critter's karma is unchanged.

2. If one Critter choses to fight and the other does not, the fighter automatically wins. Once again, the losing Critter loses 25 health, but the winning Critter *loses* karma for violently attacking a friendly Critter. The amount of karma lost depends on the action chosen by the friendly Critter: 3 karma lost for a PARTY action and 5 karma lost for a HEAL action. Again, the losing Critter's karma is unchanged.

3. If both Critters choose not to fight, both gain karma and neither loses health. In this case, a PARTY action earns 3 karma. If a Critter chooses the HEAL action and the other Critter has less than 100 health, then the other Critter's health increases by up to 50 (for a maximum of 100 health), and the Critter performing the HEAL action gains 5 karma. On the other hand, if a Critter chooses HEAL and the other Critter has full health, no healing is done and the healer gains no karma.

Each run of the environment starts with multiple instances of several species of Critters. If an individual Critter's health reaches 0 (or below), it is removed from the environment, and it can no longer accumulate karma points. Each environment is simulated (for a maximum number of iterations), and the total karma points earned by each species of Critter will be calculated.

Assignment

You can get started on the assignment by following this link:

https://classroom.github.com/a/ZiPEjjYo

After you have the source code, your assignment is to **implement four species of Critters** that will interact in the Critter environment. Your Critters should be in **Python** files named name1.py, name2.py, name3.py, and name4.py, where "name" is replaced by your first initial

and last name. So for me, my Critters would be named aeck1.py, aeck2.py, aeck3.py, and aeck4.py. Each file should contain a different Python class, where the name of the class is the same as the name of the file, except with your first initial capitalized (for me, Aeck1, Aeck2, Aeck3, Aeck4). You are allowed (and encouraged) to work with a partner for this assignment, but you should still design and implement four Critter classes per student.

To implement a new Critter species, you should inherit from the Critter class in critter.py. In particular, you need to implement 5 functions:

```
# decides what the Critter should do when interacting with
# another Critter
#
# oppInfo describes the environment
def interact(self, oppInfo):
    # your code for deciding how to interact goes here


# determines what direction the Critter should move in from the set
# critter.NORTH, critter.EAST, critter.SOUTH, critter.WEST
#
# info describes the environment
def getMove(self, info):
    # your code for deciding how to move goes here


# returns a String that should be used to display the Critter on the
# environment GUI
def getChar(self):
    # your code for choosing a single character to display for your
    # individual Critter goes here


# returns a color constant (defined in color.py) that should be used
# to display the Critter on the environment GUI
def getColor(self):
    # your code for choosing a color to display for your
    # individual Critter goes here


# alerts the Critter that an interaction happened so that it can
# update its knowledge about the environment
#
# won is a Boolean indicating whether or not the Critter won its
# interaction
# oppFight is the chosen action of the opponent
def interactionOver(self, won, oppFight):
    # your code for optionally saving and using information after an
    # interaction goes here
```

You should implement your Critter's intelligent decisions of how to behave in the environment in the `interact` and `getMove` functions, whereas `interactionOver` will help the Critter gain information about interactions that it can use to decide how to act in the future. Please be creative and have fun when giving your Critters the ability to reason about how to act!

At least one of your Critters should use some AI technique that we studied this semester (e.g., Q-Learning, MDPs, CSPs, game theory, search) as part of its decision making. You are allowed to adapt your code from any of the previous four assignments for this purpose. **At least two other Critters** should have interesting, (semi-)intelligent behavior. That is, do not *only* create really simple Critters that (1) pick the same action every time, (2) pick purely random actions each time, etc. This is your chance to have some fun, be creative, and think about how an agent (i.e., a Critter) might make its own decisions to achieve a goal (i.e., maximize its karma, possibly by needing to survive as long as possible).

Of note: Critters have internal fields `health` and `karma` storing how much health they have (from 0 to 100) and how much karma they have earned (starting at 0). These fields must **NOT** be updated by your code, but you can read them if it will help your Critter make wise decisions of how to act.

The `oppInfo` and `info` parameters to `interact` and `getMove`, respectively, are `CritterInfo` objects containing the following fields and functions:

```
x # the Critter's x coordinate
y # the Critter's y coordinate
width # the width of the simulation world
height # the height of the simulation world
char # the Critter's display character
color # the Critter's display color
getNeighbor(direction) # a function that, when called with a parameter
   # representing one of the direction constants, returns the name of
   # the class (NOT the display character) of the Critter in that
   # location (i.e. the location that is one space in the given
   # direction of the current Critter.) and "." if there is no Critter
   # there. locations include critter.NORTH, critter.NORTHEAST, etc.
getNeighborHealth(direction) # a function that, when called with a
    # parameter representing one of the direction constants, return
    # the health of the Critter in that location and 0 if there is
    # no Critter there
```

To test your Critters, you can run the Critter environment by running the following program:

```
python3 critter_main.py
```

It will automatically populate the environment with 25 individuals of every Critter subclass that it finds in the same directory as the `critter_main.py` program. I have provided you with 5 example species that make very simple decisions.

README

Within a README file, you should include:

1. Two or three sentences per Critter species explaining how your Critters make decisions and how you implemented those decision processes.
2. A short paragraph describing your experience during the assignment (what did you enjoy, what was difficult, etc.)
3. An estimation of how much time you spent on the assignment, and
4. An affirmation that you adhered to the honor code

As you implement your four Critters, please remember to commit them to your repository on GitHub. You do not need to wait until you are done with the assignment; it is good practice to do so not only after each coding session, but maybe after hitting important milestones or solving bugs during a coding session. ***Make sure to document your Critter class files***, explaining what they do and how you implemented them.

Honor Code

Each student is allowed to complete this assignment with a single partner or alone. Students are encouraged to collaborate with one another to discuss the abstract design and processes of their implementations. For example, please feel free to discuss different ideas for how Critters might make decisions, as well as help each other use Python as our programming language (for those who primarily use other languages). At the same time, since this is a pair-programming assignment, no code can be shared between groups, nor can students look at code from someone other than your partner.

Grading Rubric

Your solution and README will be graded based on the following rubric:

Four Critter classes (per student) make all requested decisions without crashing: /40 points
Critters operate efficiently without slowing down the environment: /10 points
Use of some AI technique from class: /15 points
Two other intelligent Critters: /20 points
Answers to README questions: /10 points
Appropriate code documentation: /5 points

By appropriate code documentation, I mean including a header comment at the top of each file explaining what the file provides, as well as at least one comment per function explaining the purpose of the function.