Discussion Week 10

Critters!
Critter Applet!

- [http://ieng6.ucsd.edu/~cs8s2/critters.html](http://ieng6.ucsd.edu/~cs8s2/critters.html)
- The most difficult part of this assignment is figuring out how to keep track of STATES and figuring out how all the components connect.
- Don’t forget the survey!
Files

- Critter.java
- CritterController.java
- Interactor.java
- Chaser.java
- Runner.java
- Random.java
- Custom.java
- Critters.html

Everything in **bold** is starter code given in the public folder.
Let’s look at the starter code!
Critter.java

- What does a critter know?
  - Its x, y coordinates (two global variables, x and y)
  - How to paint itself (a paint() method)
  - How to react to other critters (a reactTo() method)

- You may also want to add a method that can calculate distance between “this” and another critter.
This is your “main” file.

It implements Runnable – so you can run more than one thread.

The starter code just gives you the GUI layout and simulation code, but otherwise doesn’t do anything.
CritterController layout

Top panel:
- Please add two or more Critters
- Start, Stop, Clear buttons

Critter panel (world)

Bottom panel:
- Select which Critter to place
- Chaser, Runner, Random, Custom buttons
- Applet started.
Listeners

- Which components do you need a listener for?
- There are multiple buttons on this GUI – do you need an ActionListener for each one?
Listeners

- Top panel – ActionListener for the Start/Stop/Clear buttons.
- Critter panel – MouseListener so that we can place the critters in the world.
- Bottom panel – ActionListener for the critter buttons.
CritterController – Class Variables

- Constants that may come in handy when checking which critter to add:
  - private static final int RUNNER = 0;
  - private static final int CHASER = 1;
  - private static final int RANDOM = 2;
  - private static final int CUSTOM = 3;

- Control how quickly the simulation runs:
  - private static final int DELAY = 50;
private ArrayList<Critter> critterList;
private Interactor actor;
private boolean running;
private CritterPanel world;
private JLabel stateLabel;
private JLabel critLabel;
CritterController – Methods

- **public void init()**
  - Places all of the GUI components in the applet.
  - Starts a new thread.

- **public void run()**
  - Runs the simulation (only if running == true).
  - Makes the critters interact with each other using the Interactor.
The run() method

- Yes, this is an infinite loop!
- But the critters will only interact (i.e. simulate) when running = true.

- How does that work? Won’t we get stuck in the infinite loop??
Threads!

- Use these when you want tasks to run at the same time (i.e., in parallel).
- One thread is listening for the events, while the other is running the simulation.
- When `start()` is called, the program “clones” itself. Then, the clone starts running the `run()` method.
Interactor

1) Determine which critter to interact with in the critterList (find the closest critter).

2) Once you find the closest critter, c must react to that closest critter (call reactTo()).

3) Critters cannot interact with themselves.

How do you determine which critter is closest?

- Calculate the distance using the critters’ x and y coordinates.
Recap

- CritterController creates the GUI and spawns a new thread for running the simulation.
- The user must first add critters to the world.
- Start the simulation.
- The run() method calls the Interactor’s interact() method to make the critters interact with each other.
- The Interactor calls the critters’ reactTo() method to determine how they will react to each other.
Adding critters to the world
Critter algorithms

- **Chaser** – tries to chase (move closer to) the closest critter. Displays as a red circle.
- **Runner** – tries to run away from the closest critter. Displays as a green square.
- **Random** – moves around randomly within a 10-pixel radius. Displays as a blue ‘X’
- **Custom** – does whatever you want it to!

Critters react to other critters by calling their reactTo() method.
public void reactTo( Critter c, Rectangle r )

- Make “this” react to critter c within the bounds of rectangle r.

- **Example: Chaser**
  - Calculate the distance between “this” and critter c for each of the possible move options (x,y is where “this” is currently):

<table>
<thead>
<tr>
<th>x-1, y-1</th>
<th>x, y-1</th>
<th>x+1, y-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-1, y</td>
<td>x, y</td>
<td>x+1, y</td>
</tr>
<tr>
<td>x-1, y+1</td>
<td>x, y+1</td>
<td>x+1, y+1</td>
</tr>
</tbody>
</table>

- Pick the move that brings you closest to critter c (the one that gives you the smallest distance value)
- Set this’s new x and y coordinates to be that of the chosen move.
Various Notes

- You CAN add critters while running the simulation – this is one of the advantages of using threads.
- Chasers can chase each other.
- Don’t forget to update the status labels when you change state!