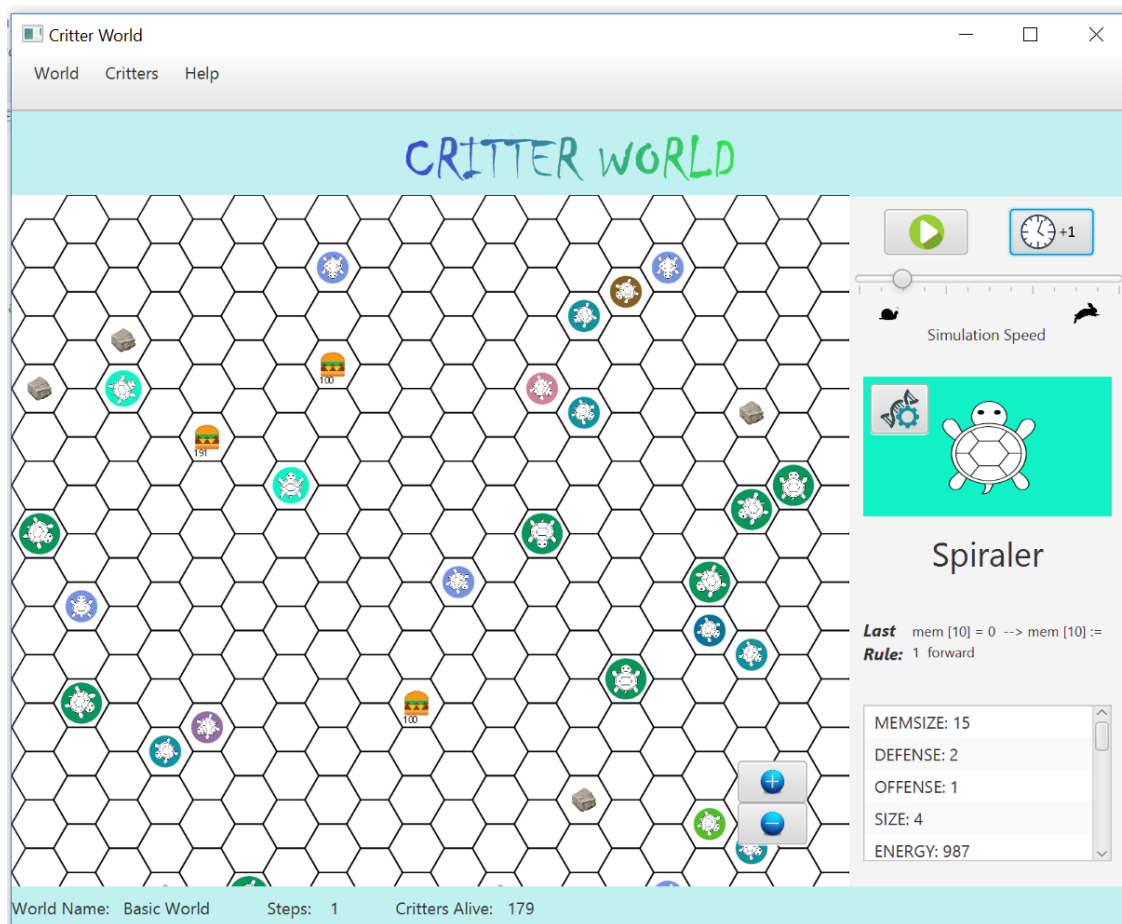


Critter World: For CS 2112 at Cornell University

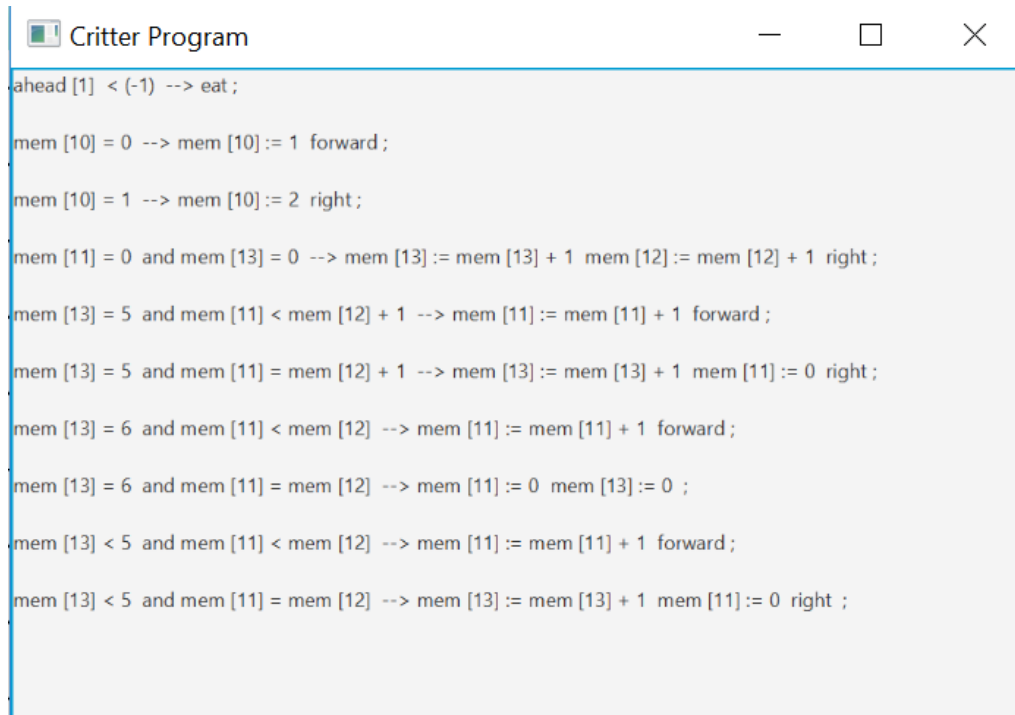
By Andrew Tsakiris and Eugene Kim

Unfortunately, I am not permitted to share the complete code for this project publicly due to academic integrity policies at Cornell University. I have included this pdf summary of the project as a replacement in this repository until actual code samples can be posted safely.

Critter World is a distributed program that simulates a world of small frogs all with one goal: survival. The critters move about, eat food, attack each other, and even mate with other compatible critters in their hexagonal world.



Each critter species is defined by a “Critter File” that contains attributes for the critter as well as a “Critter Program.” This program is written in the “Critter Language,” a context-free grammar (defined by the course staff in the project spec) for which I wrote a parser and interpreter. Armed with their programs defining their behavior, critters execute actions in their effort of survival. New critters can be formed with new, slightly mutated programs by mating and budding, which was implemented using fault injection in the abstract syntax trees constructed by the interpreter of the “Critter Language.”



```
ahead [1] < (-1) --> eat ;

mem [10] = 0 --> mem [10] := 1 forward ;

mem [10] = 1 --> mem [10] := 2 right ;

mem [11] = 0 and mem [13] = 0 --> mem [13] := mem [13] + 1 mem [12] := mem [12] + 1 right ;

mem [13] = 5 and mem [11] < mem [12] + 1 --> mem [11] := mem [11] + 1 forward ;

mem [13] = 5 and mem [11] = mem [12] + 1 --> mem [13] := mem [13] + 1 mem [11] := 0 right ;

mem [13] = 6 and mem [11] < mem [12] --> mem [11] := mem [11] + 1 forward ;

mem [13] = 6 and mem [11] = mem [12] --> mem [11] := 0 mem [13] := 0 ;

mem [13] < 5 and mem [11] < mem [12] --> mem [11] := mem [11] + 1 forward ;

mem [13] < 5 and mem [11] = mem [12] --> mem [13] := mem [13] + 1 mem [11] := 0 right ;
```

Critter World is a Java FX application and can be launched on a server with which multiple clients can interact at read, write, or admin access. We wrote our server and client using Spark, HTTP, and JSON.

We used a modified version of Dijkstra's Shortest Path Algorithm to create nearest food sensing capabilities for the critters. Critters can use a "smell" command to locate the nearest food source (algorithm returns the number of moves needed to reach the food as well as the relative direction the critter should move in).

Hopefully I will be able to post code samples soon, but until then please feel free to contact me through my website andrewtsakiris.com if you wish to discuss this project further and see code samples.