Taylor Eldrith, Mark Piccirilli, Amy Sage, Andrew Spalla

CS162

18 February 2018

Group Project Design and Reflection

The intent of this project is to design a Predator-Prey simulation consisting of a predator "Doodlebug: class and a prey "Ant" class. These two classes are child classes to the "Critter" parent class, and are polymorphic in nature with respect to two particular functions: move and breed. An additional function will be used within the doodlebug class to starve off objects that do not eat an ant within the allotted time. There are several project requirements associated with the proposed simulation. Between the two critter child classes, the doodlebug objects will randomly move about the 2D grid before the ant objects. During this movement, doodlebugs will be able to randomly eat an adjacent ant if available. Following doodlebug movement, ant objects will move about the board next. Following critter movement, all critter objects will be advanced in age by a single time step and next be assessed if breeding can occur for the respective class objects. After these events, an iteration is complete and assessed against the remaining user requested iterations. Finally, each iteration along the grid is to be output in ASCII characters to demonstrate the successive iterations.

Several design choices were considered for this project. Project requirements specified that the Critter class should have a virtual function called move that is defined in the derived classes of Ant and Doodlebug. Due to similarities, it was assessed that the Breed function could be derived in a similar fashion.

Test Plan

Test Case	Input Values	Expected Outcome	Observed Outcome
Unconstrained Ant on Grid	Single instantiated ant object	Ant Moves Up/Down/Left/Right	Observed results matched expected outcome
Ant bound by top border	Single instantiated ant object	Ant cannot move Up, but moves Down/Left/Right	Observed results matched expected outcome
Ant bound by bottom border	Single instantiated ant object	Ant cannot move Down, but moves Up/Left/Right	Observed results matched expected outcome

Ant bound by right border	Single instantiated ant object	Ant cannot move Right, but moves Up/Down/Left	Observed results matched expected outcome
Ant bound by left border	Single instantiated ant object	Ant cannot move Left, but moves Up/Down/Right	Observed results matched expected outcome
Ant ready to breed	Ant is 3 times steps old	Ant will randomly breed in an adjacent spot	Observed results matched expected outcome
Ant ready to breed	Ant is 3 times steps old, but all adjacent spots are taken.	Ant will not breed.	Observed results matched expected outcome
Unconstrained Doodlebug on Grid	Single instantiated Doodlebug object	Doodlebug Moves Up/Down/Left/Right	Observed results matched expected outcome
Doodlebug bound by top border	Single instantiated Doodlebug object	Doodlebug cannot move Up, but moves Down/Left/Right	Observed results matched expected outcome
Doodlebug bound by bottom border	Single instantiated Doodlebug object	Doodlebug cannot move Down, but moves Up/Left/Right	Observed results matched expected outcome
Doodlebug bound by right border	Single instantiated Doodlebug object	Doodlebug cannot move Right, but moves Up/Down/Left	Observed results matched expected outcome
Doodlebug bound by left border	Single instantiated Doodlebug object	Doodlebug cannot move Left, but moves Up/Down/Right	Observed results matched expected outcome
Doodlebug surrounded by Ant objects	1x Doodlebug, 4x Ants	Doodlebug randomly eats an adjacent ant and moves to that location	Observed results matched expected outcome
Doodlebug ready to breed	Doodlebug is 8 times steps old	Doodlebug will randomly breed in an adjacent spot	Observed results matched expected outcome
Doodlebug ready to breed	Doodlebug is 8 times steps old, but all adjacent spots are	Doodlebug will not breed.	Observed results matched expected outcome

	taken.		
Doodlebug starves	Doodlebug is 3 time steps old, and has not eaten ant	Doodlebug dies	Observed results matched expected outcome

Group Work Distribution

Group members spent ample time brainstorming the various material in regards to constructing an effective solution to this program. After determining several design choices based on the project requirements, program implementation focused on utilizing the necessary topics covered in the course. Each member was integral in the final product, and everyone made worthwhile and meaningful contributions towards the writing, styling and coding of the source files.

Reflection

While writing the source files, several items occurred that required alterations to our original plan. One design choice was to consider the case where breeding is unsuccessful. If it was an unsuccessful breed attempt, we decided to still require the critter to complete the necessary number of timesteps between breeding events. Another factor that occurred was the implementation of numerical constraints that were implemented with the extra credit. It was necessary to restrict the number of rows and columns of the board to a reasonable size, and start with an appropriate number of ants and doodlebugs that will never be larger than the size of the board. While completing this project, the group learned to consolidating several topics that were integral in previous assignments. For example, the melding of current topics (Inheritance and Polymorphism) with previously introduced material of dynamic allocation and pointer implementation to create a much more involved and efficient programs. The group also got some experience working with other people on a programming assignment, and learned that organizing the workload and communicating can be difficult. However, working in groups is key to any kind of software development, and is vital skill in this field.